

P-15971

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO

DIRECCIÓN DE NUEVAS CREACIONES

E. S. D.

Re: Expediente No. **NC2018/0008835**
Solicitud de patente a nombre de **ICAT LLC**

Yo, CAROLINA DAZA MONTALVO identificada como aparece al pie de mi firma, domiciliada en la Carrera 11 No. 86-53, obrando como apoderada de la sociedad ICAT LLC, domiciliada en Indiana, Estados Unidos de América, solicitante de la patente de la referencia, en respuesta al oficio No. **3023** notificado mediante comunicación electrónica el 3 de marzo de 2020, relacionado con concepto técnico de fondo, me permito manifestar:

1. Modificación a las Reivindicaciones

Con fundamento en la facultad que confiere el artículo 34 de la Decisión 486 al solicitante para modificar la solicitud de patente en cualquier momento del trámite, presento junto con este memorial nuevo capítulo reivindicatorio, el cual consta de doce (12) reivindicaciones y reemplaza completamente el capítulo presentado anteriormente. En el nuevo capítulo reivindicatorio se realizaron las siguientes modificaciones:

- 1.1** Hemos modificado la antigua reivindicación 1 para ahora incluir la característica "(...) *que precompila código fuente y separa las instrucciones al núcleo canalizado algorítmico reutilizable, (...) genera código para configurar el núcleo canalizado algorítmico reutilizable, identifica a partir del código fuente algoritmos matemáticos y lógicos capaces de optimizarse mediante una configuración del núcleo canalizado algorítmico reutilizable, (...) y el compilador canalizado de coincidencia algorítmica genera un código para la unidad de procesamiento que le indica a la unidad de procesamiento que configure el núcleo canalizado algorítmico reutilizable, en el que el núcleo canalizado algorítmico reutilizable es reconfigurable*" el debido soporte se encuentra en la

en los párrafos 23, 30 y 34 de la solicitud internacional y en la descripción original presentada y así dar claridad a la invención reclamada.

- 1.2** Se ha eliminado la antigua reivindicación 5, la cual hacía referencia a *"las puertas programables de campo están configuradas para completar tareas sin más sobrecarga desde la unidad de procesamiento"* y así dar claridad a la invención.
- 1.3** Se ha modificado la antigua reivindicación 11 para ahora recitar "un sistema que comprende una pluralidad de núcleos según la reivindicación 1, que comprenden un procesador para ejecutar los pasos de ..." y así dar claridad a la materia reclamada.

En cuanto a estas modificaciones, me permito decir que las mismas no amplían de ninguna manera la materia objeto de la presente solicitud y se realizaron con el ánimo de superar las objeciones de claridad y patentabilidad planteadas por el examinador en su concepto técnico.

2. Descripción (Art. 28 D486)

Es deseo del solicitante presentar un nuevo capítulo descriptivo para corregir los errores de traducción de la presente solicitud, en particular en los párrafos 0070 y 0079, de acuerdo a las sugerencias del examinador.

3. Pago de tasas

En relación con el número de reivindicaciones, atentamente nos permitimos decir que no se adjunta tasa por reivindicaciones adicionales, toda vez que el nuevo capítulo reivindicatorio contiene el mismo número de cláusulas que el anteriormente presentado.

Por otra parte, considerando que el presente memorial se presenta con modificaciones a las reivindicaciones, se adjunta recibo por concepto de examen de patentabilidad, según lo establecido en la Resolución 3719 del 2016.

4. Novedad/Nivel inventivo

El señor examinador establece en su concepto técnico la falta de novedad de las antiguas reivindicaciones 1 a 7, 9 y 10 a la luz de las enseñanzas técnicas descritas en el documento US 2003/0066057 (D1) y la falta de nivel inventivo de las antiguas reivindicaciones 8 y 11 a 21 a la luz de la combinación de las enseñanzas técnicas descritas en los documentos (D1) y US2014/0040552 (D2).

A este respecto, me permito decir que el artículo 16 de la Decisión 486 de la Comisión de la Comunidad Andina establece que la invención se considera nueva cuando no está comprendida en el estado de la técnica. Si de acuerdo con el espíritu de la norma legal indicada, la invención no se considera como nueva si está "comprendida" en el estado de la técnica, dicha norma se está refiriendo, naturalmente, a la totalidad del invento y no a alguno o a algunos de los componentes.

Aplicando lo anterior al caso concreto, para que las razones que aduce ese despacho para desvirtuar la novedad fuesen ciertas, sería forzoso que se demostrara que la invención de mi mandante esté comprendida en el estado de la técnica, específicamente en forma individual en los documentos D1 y/o D2 de forma TOTAL, lo cual no sucede con la materia reclamada en el nuevo capítulo reivindicatorio.

Por otra parte, me permito decir que el artículo 18 de la Decisión 486 de la Comisión de la Comunidad Andina establece que *"Se considerará que una invención tiene nivel inventivo, si para una persona del oficio normalmente versada en la materia técnica correspondiente, esa invención no hubiese resultado obvia ni se hubiese derivado de manera evidente del estado de la técnica"*.

Lo anterior significa que el análisis de nivel inventivo **NO** es una comparación de los objetos de las anterioridades y la solicitud en estudio. El análisis de nivel inventivo **debe demostrar** la obviedad de la invención y establecer cómo ésta no contiene un salto cualitativo, que demuestre su altura inventiva, teniendo en cuenta que se hace **necesario demostrar** que la invención analizada carece de nivel inventivo y hubiese sido obvia para una persona versada en la materia.

Considerando las anteriores definiciones, es deseo de la solicitante manifestar su desacuerdo con dichas objeciones y presentar argumentos que demuestran la novedad de la invención que se reclama en las nuevas reivindicaciones 1 a 23.

En cuanto a la Novedad de la invención, apreciamos los comentarios del examinador, pero respetuosamente no estamos de acuerdo, ya que, como se muestra a

continuación, la presente invención es novedosa sobre el arte previo citado como D1 y D2.

En relación con la Novedad de la presente invención, al menos las características agregadas a la reivindicación 1, que se indican anteriormente, se consideran novedosas sobre el documento D1.

El documento D1 permanece en silencio con respecto a las características técnicas: *"un compilador canalizado de correlación algorítmica, que precompila código fuente y separa las instrucciones al núcleo canalizado algorítmico reutilizable, de modo que el compilador canalizado de correlación algorítmica precompila el código fuente diseñado para funcionar en un procesador estándar sin procesamiento paralelo para procesamiento por la unidad de procesamiento ... genera código para configurar el núcleo canalizado algorítmico reutilizable, identifica a partir del código fuente algoritmos matemáticos y lógicos capaces de optimizarse mediante una configuración del núcleo canalizado algorítmico reutilizable ... y el compilador canalizado de coincidencia algorítmica genera un código para la unidad de procesamiento que le indica a la unidad de procesamiento que configure el núcleo canalizado algorítmico reutilizable, en el que el núcleo canalizado algorítmico reutilizable es reconfigurable"*.

Aunque el documento D1 identifica que FPGA puede configurarse como un proceso paralelo canalizado que puede usarse para ejecutar bucles, no sugiere de ninguna manera que un código fuente que sea para un procesador estándar sin procesamiento paralelo se recompile para identificar algoritmos para procesamiento paralelo.

Por lo tanto, la reivindicación 1 y todas las reivindicaciones que dependen de ella son nuevas sobre el documento D1.

Ahora bien, en relación con el nivel inventivo de la presente solicitud. Un efecto técnico que tiene la nueva característica mencionada anteriormente es que permite que el código escrito para un núcleo de procesamiento no paralelo se ejecute de una manera más eficiente sin volver a escribir sustancialmente el código de alto nivel como se describe en la solicitud internacional tal como se presentó en el párrafo 20.

Por lo tanto, el problema técnico objetivo puede formularse como, cómo ejecutar el código de programa de computadora existente cuando las capacidades de hardware han mejorado.

Ante el problema técnico objetivo y a partir del documento D1, una persona experta aprendería a utilizar compiladores vectorizados. Las transformaciones de bucle se usan en el documento D1 para personalizar las tuberías para cumplir con las restricciones de recursos de hardware mientras se maximiza el paralelismo disponible.

Sin embargo, el documento D1 no indica cómo se podría ejecutar un código fuente existente con hardware mejorado sin esencialmente volver a escribir el código.

Por lo tanto, la persona experta no podría resolver el problema técnico objetivo basándose en el documento D1.

La solución del documento D1 introduce precompilación e identificación a partir del código fuente de algoritmos matemáticos y lógicos capaces de optimizarse mediante una configuración del núcleo canalizado algorítmico reutilizable. Como se mencionó, el documento D1 no ofrece pistas hacia una solución que se asemeje al enfoque de la reivindicación 1. Por lo tanto, la solución de la reivindicación 1 no habría sido obvia para las personas expertas que intentan resolver el problema técnico objetivo basándose en el documento D1.

Por lo tanto, la reivindicación 1 y todas las reivindicaciones que dependen de ella tienen un nivel inventivo.

Con base en el análisis anterior, ni D1 ni D2 enseñan o sugieren la presente invención. En vista de lo anterior, las reivindicaciones presentadas son novedosas e inventivas sobre el estado del arte. Por lo tanto, se le solicita respetuosamente al señor examinador retire la objeción de novedad y nivel inventivo.

De esta manera, queda demostrado que los documentos D1 y/o D2 citados por su Despacho **no puede ser considerados como anterioridades válidas** capaces de afectar la novedad y el nivel inventivo de la solicitud, toda vez que estos documentos fallan en enseñar todas las características esenciales de la invención, y la manera como se solucionó el problema técnico en cuestión.

En este sentido, y siguiendo el Manual Andino de Patentes, la presente invención tal y como se encuentra definida en las reivindicaciones 1 a 23, tiene diferencias con el estado del arte e implicó un nivel inventivo durante su desarrollo, por lo cual debe ser considerada como novedosa e inventiva según las disposiciones de los artículos 16 y 18 de la Decisión 486.

En consecuencia, solicitamos a ese despacho pronunciarse en relación con todos los argumentos presentados en este memorial dando estricta aplicación al principio del debido proceso consagrado en el artículo 29 de la Constitución Nacional.

5. Conclusión

Con las modificaciones introducidas y los argumentos presentados puede concluirse que:

- i) La presente solicitud es clara, concisa y está debidamente soportada en la descripción, cumpliendo con los requerimientos del Artículo 30 de la Decisión 486; y
- ii) Las reivindicaciones actuales 1 a 23 son patentables sobre D1 y/o D2, cumpliendo así con los requerimientos de los Artículos 16 y 18 de la Decisión 486.

En consecuencia, se solicita atentamente otorgar el privilegio de patente solicitado, toda vez que esta solicitud cumple con los requerimientos de la Decisión 486.

Habiendo atendido cabalmente las objeciones presentadas por ese despacho, atentamente solicitamos se conceda el privilegio de patente de invención a la solicitud en estudio o si el señor examinador considera que subsisten razones para objetar la presente invención, rogamos expedir otro examen de fondo con los debidos motivos y razones que sustenten su objeción con el fin de que nuestro mandante tenga la posibilidad de exponer sus argumentos con base en la interpretación de los artículos 45 de la Decisión 486 -el cual otorga la facultad al examinador para emitir los exámenes de fondo- y 42 del Código de Procedimiento Administrativo y de lo Contencioso Administrativo que señala que: "*Habiéndose dado oportunidad a los interesados para expresar sus opiniones, y con base en las pruebas e informes disponibles, se tomará la decisión, que será motivada. La decisión resolverá todas las peticiones que hayan sido oportunamente planteadas dentro de la actuación por el peticionario y por los terceros reconocidos*", (el subrayado es nuestro).

Finalmente, solicitamos a ese Despacho pronunciarse en relación con todos los argumentos presentados en este memorial dando estricta aplicación al principio del debido proceso consagrado en el artículo 29 de la Constitución Nacional.

6. Anexos

- Capítulo reivindicatorio que incluye veintitrés (23) reivindicaciones.
- Nuevo Capítulo Descriptivo modificado
- Recibo de pago por un nuevo examen de patentabilidad

Señor Director,



CAROLINA DAZA MONTALVO

C.C. 66.783.790 de Palmira Valle

TP-141563 CSJ

notificaciones@clarkemodet.com.co

**PROCESADOR CON NÚCLEO CANALIZADO ALGORITMICO RECONFIGURABLE
Y COMPILADOR CANALIZADO DE CORRELACION ALGORITMICA**

SOLICITUDES CRUZADAS RELACIONADAS

5 **[0001]** Esta solicitud reivindica la prioridad de la
Solicitud Provisional de los Estados Unidos 62/287,265
titulada Procesador con Núcleo Canalizado Algorítmico
Reconfigurable y Compilador Canalizado de Correlación
Algorítmica, que se archivó el 26 de enero de 2016, que se
10 incorpora aquí en su totalidad, como referencia.

CAMPO DE LA INVENCION

[0002] El campo se relaciona a la programación de
computadoras y diseño y programación de microprocesadores,
especialmente el procesamiento reconfigurable, canalizado y
15 paralelo de las instrucciones del software de propósito
general.

ANTECEDENTES

[0003] La Figura 1A ilustra un compilador de procesador
convencional. Los procesadores convencionales, como los
20 microprocesadores Intel y los microprocesadores ARM son bien
conocidos. Por ejemplo, una ilustración conceptual de un
procesador convencional se muestra en la Figura 1B. Estos
procesadores son el corazón de las unidades centrales de
procesamiento para computadoras y dispositivos modernos y se
25 utilizan para procesar algoritmos. Un problema con los
procesadores convencionales es que estos tipos de

procesadores son de uso general y no son reconfigurables de ninguna manera práctica que permita mejorar su rendimiento para aplicaciones específicas. Otro problema es que el control de ejecución del programa agrega una sobrecarga

5 sustancial al procesamiento de funciones algorítmicas, tales como operaciones matemáticas y decisiones lógicas que modifican el flujo de procesamiento. Se puede usar un lenguaje de programación de nivel superior para programar el procesador convencional, y el compilador convierte las

10 instrucciones en el lenguaje de programación de nivel superior en código de máquina para la arquitectura de procesador particular. Este código de máquina se proporciona a una ubicación de memoria accesible por el procesador y proporciona instrucciones para el funcionamiento del

15 hardware del procesador, junto con cualquier BIOS u otras llamadas proporcionadas por la arquitectura del sistema. En la mayoría de los casos, las instrucciones matemáticas y de procesamiento lógico se dirigen a una unidad lógica aritmética (ALU), que devuelve una solución a la parte de

20 control de ejecución del programa del procesador, que gestiona por encima, como guiar al procesador en el orden correcto de resolución matemática algoritmos, decisiones lógicas, manejo de datos y similares.

Las instrucciones del código de máquina se extraen continuamente del almacenamiento del programa para controlar el procesamiento de los datos. Esta sobrecarga limita significativamente el rendimiento de la máquina.

5 **[0004]** Por ejemplo, lo siguiente ilustra los pasos de un compilador convencional que compila una operación matemática en un lenguaje de programación "C", que es un ejemplo de un lenguaje de programación de nivel superior que puede compilarse para crear código de máquina para un procesador

10 convencional particular. Una operación matemática simple asigna "var i1;" "var i2"; y "var s" para definir una ubicación de almacenamiento de datos para la variable i1, i2 y resultado s. Entonces, una instrucción "s = i1 + i2;" se puede usar para sumar las variables asignadas en las

15 ubicaciones de datos i1 e i2. El compilador (a) primero asigna ubicaciones de almacenamiento para los datos (por ejemplo, i1, i2 y s) y (b) genera el código fuente en el código máquina. Un procesador convencional recuperaría todo o parte del código de máquina desde una ubicación de memoria

20 en la que se almacena el código. Luego ejecutaría el código de máquina. Para este ejemplo, la unidad de procesamiento central (CPU) cargaría los datos i1 en una ubicación de memoria y lo enviaría a la ALU, cargaría los datos i2 en una ubicación de memoria y los enviaría a la ALU, e instruiría

25 a la ALU para que agregue los datos ubicados en i1 e i2. Solo entonces la ALU realizará una adición de los valores

ubicados en las ubicaciones de datos para i1 e i2. Este es el paso de trabajo útil, con la configuración de la CPU por encima. Luego la CPU podría obtener el resultado ALU de la ubicación de datos para "s" y podría enviarlo al controlador de entrada y salida. Este es un paso necesario para presentar el resultado, si el resultado no es un paso intermedio en el cálculo. Los procesadores convencionales evolucionaron a partir del deseo de ahorrar tiempo en el desarrollo de programas informáticos, permitiendo la compilación de lenguajes de programación de mayor nivel para diversas arquitecturas de unidades centrales de procesamiento e instalaciones periféricas. Además, todos los procesos ejecutados por la CPU pueden compartir una ALU común, compartiendo por tiempo la ALU entre varios programas que operan en el entorno del sistema.

[0005] Se conocen circuitos integrados específicos de aplicación (ASICs) que se integran en circuitos electrónicos de hardware capaces de realizar rápidamente cálculos para funciones específicas. Estos reducen la sobrecarga mediante el cableado de funciones específicas en el hardware.

[0006] Se conocen algunos arreglos de puertas programables de campo (FPGAs) que tienen un gran número de puertas lógicas y bloques de memoria de acceso aleatorio (RAM). Estos FPGAs son usados para implementar cálculos digitales complejos. Dichos diseños de FPGA pueden emplear buses de datos de entrada/salida y bidireccionales muy

rápidos, pero es difícil verificar el tiempo correcto de los datos válidos dentro del tiempo de configuración y tiempo de espera. La planificación del piso permite que las asignaciones de recursos dentro de los FPGAs cumplan con estas limitaciones de tiempo. Los FPGAs se pueden usar para implementar cualquier función lógica que un chip ASIC podría realizar. La capacidad de actualizar la funcionalidad después del envío, la reconfiguración parcial de una parte del diseño y un bajo costo de ingeniería no recurrente en relación con un diseño ASIC ofrece una ventaja para algunas aplicaciones, incluso cuando se toma en consideración el costo unitario generalmente más alto.

[0007] Sin embargo, la penetración de arquitecturas FPGA se ha limitado a productos de nicho estrecho. Una computadora virtual FPGA para ejecutar una secuencia de instrucciones de programa mediante la reconfiguración sucesiva de un grupo de FPGA en respuesta a esas instrucciones fue patentada en la patente de los EE. UU. N° 5,684,980. La Figura 2 ilustra una estructura para esta arquitectura FPGA. Esta patente emitida incluye un arreglo de FPGA que cambia configuraciones de forma sucesiva durante la ejecución de sucesivos algoritmos o instrucciones. La configuración de un arreglo de FPGAs permite que se ejecute un algoritmo completo o un conjunto de instrucciones sin esperar a que se descargue cada instrucción en cada paso computacional.

[0008] Los desarrollos en FPGAs y la integración con procesadores prometen la capacidad de reprogramarse en "tiempo de ejecución", pero en realidad, la computación reconfigurable o los sistemas reconfigurables para adaptarse a la tarea en cuestión están lejos de ser implementados en aplicaciones prácticas debido a las dificultades en la programación y configuración de estas arquitecturas para este propósito.

[0009] La Figura 2 ilustra un diagrama de bloques de una computadora virtual que incluye un arreglo de arreglos de puertas programables de campo y dispositivos de interconexión programables de campo (FPIN) o interruptores de barra cruzada que liberan recursos internos de los arreglos de puertas programables de campo de cualquier tarea de conexión externa, como se describe en la patente de los EE.UU. Número 5,684,980, cuya descripción y dibujos se incorporan aquí en su totalidad con el fin de divulgar el conocimiento de un experto en la técnica, familiarizado con los FPGAs.

[0010] La figura 2 ilustra un arreglo de arreglos de puertas programables de campo y dispositivos de interconexión programables de campo que están dispuestos y empleados como un coprocesador para mejorar el rendimiento de una computadora central o dentro de un procesador de computadora virtual para realizar algoritmos sucesivos. Los algoritmos sucesivos deben programarse para corresponderse

con una serie de instrucciones convencionales que normalmente se ejecutarían en un microprocesador convencional. Entonces, la tasa de realización de la tarea computacional específica de los algoritmos sucesivos por el
5 arreglo FPGA/FPIN es mucho menor que la velocidad de las instrucciones correspondientes realizadas por un microprocesador convencional. La computadora virtual de la Figura 2 debe incluir una sección de control reconfigurable que rija la reconfiguración del arreglo FPGA/FPIN. Los
10 archivos de bits de configuración se deben generar para la sección de control reconfigurable utilizando un paquete de software diseñado para tal fin. Luego, el archivo de bits de configuración debe transmitirse a un arreglo FPGA/FPIN correspondiente en la computadora virtual. La Figura 2
15 ilustra cómo los arreglos y la memoria de acceso aleatorio (RAM) de dos puertos están conectados por pines a la sección de control reconfigurable, una interfaz de bus y la memoria principal de la computadora. La interfaz del bus está conectada a un bus del sistema.

20 **[0011]** La Patente de EE.UU. N° 5,684,980 muestra cómo los pines proporcionan un pin de reloj y un pin que conecta la sección de control reconfigurable a los arreglos FPGA/FPIN, y muestra un ejemplo de una sección de control reconfigurable.

25 **[0012]** La Patente de EE.UU. N° 4.291.372 revela un sistema de microprocesador con un formato de instrucción

especializado que funciona junto con un módulo lógico dependiente de la aplicación externa que maneja requisitos específicos para la transferencia de datos hacia y desde un dispositivo periférico. El microprocesador proporciona una memoria de programa que tiene un formato de instrucción especializado. El formato de palabra de instrucción proporciona un campo de bit único para seleccionar un contador de programa o un registro de referencia de memoria como fuente de dirección de memoria, un campo de función que define la ruta de transferencia de datos a realizar y un campo de origen y destino para direccionar fuente y ubicaciones de destino. Anteriormente, las unidades de controlador periférico cargaban el sistema con el procesador y los circuitos de control en el módulo base para manejar los requisitos específicos.

[0013] Las unidades o arreglos DSP (Procesamiento de Señal Digital) de procesadores DSP pueden integrarse en arreglos paralelos que optimizan el rendimiento de algunas tareas gráficas intensivas, como el procesamiento de píxeles para generar imágenes en pantallas de salida, como monitores y televisores. Estos están hechos a medida e incluyen un BIOS específico para el entorno de aceleración gráfica creado para que los procesadores de señales digitales hagan su trabajo.

[0014] La conmutación de bus de matriz (MBS) es conocida. Por ejemplo, la guía del usuario "AMBA® 4 AXI4™, AXI4-Lite™

y Afirmaciones de protocolo AXI4-Stream™, Revisión: r0p1, Guía del usuario", derechos de autor 2010, 2012, denominado ARM DUI 0534B, 1D072312, enseña un sistema para conmutación de bus de matriz que es de alta velocidad e implementable por una persona con experiencia ordinaria en la técnica. La guía del usuario está escrita para diseñadores de sistemas, integradores de sistemas e ingenieros de verificación que desean confirmar que un diseño cumple con un protocolo AMBA 4 relevante. Esto puede ser AXI4, AXI4-Lite o AXI4-Stream, por ejemplo. Todas las marcas comerciales son marcas registradas de ARM en la UE y en otros lugares. Cuando se exceptúa, esta referencia se incorpora aquí en su totalidad como referencia. Un MBS es un bus de alta velocidad para la entrada y salida de datos, y esta referencia enseña los métodos y el hardware para que un ingeniero de sistemas integre un ejemplo de un MBS en una arquitectura de sistema de procesador.

[0015] Todo esto es conocido en la técnica, pero ningún ejemplo en la técnica anterior elimina casi toda la sobrecarga generada por los sistemas de procesamiento convencionales, mientras que mantiene la flexibilidad de procesar una amplia gama de algoritmos y el uso de un lenguaje de programación de nivel superior estándar, como "C", para el desarrollo de software para el sistema de procesamiento.

RESUMEN

[0016] Un procesador paralelo canalizado en un chip comprende una unidad de procesamiento y un arreglo de puertas reconfigurables, programables por campo, programadas por un compilador canalizado de correlación algorítmica, que puede
5 ser un precompilador, de modo que el compilador canalizado de correlación algorítmica precompila el código fuente diseñado para funcionar en un procesador estándar sin procesamiento paralelo para el procesamiento por parte de la unidad de procesamiento, y la unidad de procesamiento y el
10 compilador canalizado de correlación algorítmica (denominado AMPC o ASML) configuran las puertas programables de campo para operar como procesadores paralelos canalizados. Por ejemplo, el procesador se puede referir como un núcleo canalizado algorítmico reutilizable (RAPC). Los procesadores
15 paralelos están configurados para completar tareas sin más sobrecarga desde la unidad de procesamiento, como la sobrecarga para controlar una unidad de procesamiento aritmético.

[0017] En un ejemplo, un procesador canalizado
20 algorítmico reutilizable comprende un conjunto de computadoras configuradas para procesar algoritmos en paralelo utilizando lenguajes de software de nivel superior estándar, tales como "C", "C ++" o similares. Por ejemplo, el grupo de computadoras se reprograma para ejecutar
25 diferentes algoritmos según sea necesario para un cálculo

particular, en función de la salida del AMPC, que se configura con los recursos de RAPC disponibles para esto.

[0018] Por ejemplo, un núcleo canalizado algorítmico reutilizable (RAPC) puede estar compuesto por tres módulos:

5 un controlador de bus inteligente o procesador de decisión lógica (LDP), un procesador de señal digital (DSP) y un conmutador de bus de matriz. Un procesador de decisión lógica (LDP) comprende funciones lógicas reconfigurables, reprogramables según la necesidad, para controlar un

10 interruptor maestro de bus (MBS). Un DSP comprende un procesador matemático reconfigurable para realizar operaciones matemáticas. En un ejemplo, todas las operaciones matemáticas procesadas por el RAPC son procesadas por el DSP. En un ejemplo, todas las funciones

15 lógicas procesadas por el RAPC son procesadas por el LDP. Un enrutador o conmutador de bus de matriz (MBR o MBS) se define como un circuito programable reconfigurable que enruta datos y resultados de un RAPC a otro RAPC y desde/hacia un controlador de entrada/salida, y/o interrumpe los

20 generadores, según sea necesario, para completar un algoritmo, sin ninguna intervención adicional de un procesador central o periférico durante el procesamiento del algoritmo. Por lo tanto, la sobrecarga se reduce mucho mediante la canalización en comparación con el hardware

25 estático y no configurable, que requiere la intervención de un procesador central o procesador periférico para dirigir

los datos y los resultados dentro y fuera de las unidades de procesamiento aritmético. En un ejemplo, el LDP procesa las decisiones lógicas y los bucles iterativos y la memoria de resultados es proporcionada por el LDP para algoritmos de aprendizaje.

[0019] En un ejemplo, todas las operaciones matemáticas procesadas por el RAPC son procesadas por el DSP, y todas las funciones lógicas son procesadas por el LDP. En un ejemplo, una pluralidad de RAPCs se configuran como un conjunto de núcleos y cada uno de los grupos de núcleos se puede reconfigurar solo programando, sin ningún cambio en el hardware. Por ejemplo, todos los RAPCs pueden configurarse para procesar algoritmos en paralelo. En un ejemplo, el LDP usa bloques de memoria como Tablas de Búsqueda (LUT) y registros para constantes o valores aprendidos. Se puede usar un LUT de n bits para codificar cualquier función lógica booleana de n entradas como tablas de verdad usando el LUT configurado por el LDP.

[0020] En un ejemplo, un compilador canalizado de correlación algorítmica (AMPC) genera código de máquina desde un lenguaje de software compilable de nivel superior, como "C", "C ++", Pascal, Basic o similares. El AMPC puede procesar el código fuente estándar, escrito para un procesador de computadora convencional, no reconfigurable y sin derivación, para generar un código de máquina para configurar uno o más de los RAPCs. Por ejemplo, AMPC genera código de máquina a

partir de un código preexistente estándar para un procesador ARM convencional o un procesador Intel convencional, y el código máquina generado por este precompilador AMPC utiliza un procesador ARM o un procesador Intel para configurar los

5 RAPCs. Por lo tanto, un nuevo sistema informático comprende un procesador convencional, tal como un procesador ARM existente, procesador Intel, procesador AMD o similar, y una pluralidad de RAPCs, cada RAPC comprendiendo un DSP, LDM y MBS, por ejemplo. Sin embargo, a diferencia de los

10 coprocesadores o aceleradores existentes, los RAPC no son simplemente coprocesadores periféricos. En su lugar, los RAPCs se reconfiguran para resolver de forma independiente complejos algoritmos matemáticos y lógicos sin la intervención adicional del procesador convencional, después

15 de que el precompilador o AMPC configura los RAPCs para hacer su trabajo. Los valores se ingresan en el RAPC configurado y se envía una solución al MBS. En un ejemplo, una pluralidad de RAPCs está dispuesta en un solo chip, tal como un ASIC reconfigurable. Un ASIC reconfigurable significa un chip

20 diseñado para comprender RAPCs de manera que cada uno de los RAPCs es reprogramable para operaciones específicas por un AMPC y una arquitectura de procesador de propósito general existente, como un procesador ARM, un procesador AMD y un procesador Intel o similar. De esta forma, un ASIC

25 reconfigurable de este tipo puede contener 2000 RAPCs y puede operar 360 billones de instrucciones por segundo con una

velocidad de reloj de 500 MHz. Por lo tanto, un único ASIC reconfigurable que comprende 2000 RAPCs puede operar 100 veces más rápido que cualquier procesador convencional de propósito general en la actualidad. Todos los RAPCs pueden
5 operar en una configuración en línea, en paralelo, mientras los datos estén disponibles. Un único RAPC puede ejecutar instrucciones 100 veces más rápido que un procesador estándar. Un ASIC reconfigurable que comprende 20 RAPCs, que funciona a una velocidad de reloj de 500 MHz, puede ejecutar
10 30 mil millones de instrucciones por segundo. Un único chip puede comprender hasta 2000 RAPCs en un ASIC de tamaño convencional. Por lo tanto, un ASIC de tamaño convencional que comprende 2000 RAPCs puede ejecutar instrucciones 200,000 veces más rápido que un sistema de procesamiento
15 convencional, sin tener que recurrir a lenguajes de programación especializados. En cambio, los programas existentes pueden transferirse para operar con un ASIC reconfigurado que comprende una pluralidad de RAPCs y beneficiarse de la ejecución de instrucciones canalizadas,
20 en paralelo, sin reescribir sustancialmente la programación de alto nivel existente. En un ejemplo, el AMPC precompila el código existente para una arquitectura de procesador de propósito general ARM que está incorporada en un ASIC reconfigurable que comprende una pluralidad de RAPCs. Esta
25 nueva arquitectura de procesador (ICAT) logra un rendimiento sorprendente e inesperado al combinar la arquitectura del

procesador ARM y una pluralidad de RAPCs en un chip. El procesador ARM integrado en el chip ICAT ejecuta instrucciones de código de máquina generadas por AMPC a partir de programas preexistentes escritos en un lenguaje de programación de alto nivel, como "C", que configuran la pluralidad de RAPCs en el chip ICAT para ejecutar instrucciones por segundo sorprendentemente rápidas. El procesador ARM también controla la supervisión, el diagnóstico y las comunicaciones inteligentes con periféricos externos al chip ICAT. Por lo tanto, para el mundo exterior, el chip ICAT parece ser un procesador ARM muy rápido que no requiere un coprocesador matemático.

[0021] En un ejemplo alternativo, el chip ICAT puede incorporar un procesador AMD y puede parecer al mundo exterior como un procesador AMD.

[0022] En otro ejemplo más, el chip ICAT puede incorporar un procesador Intel y aparece en el mundo exterior como un procesador Intel.

[0023] Sorprendentemente, aunque el chip ICAT parece ser un procesador estándar, no reconfigurable y no canalizado, capaz de ejecutar instrucciones a un ritmo simplemente equivalente al procesador estándar, el chip ICAT ejecuta las instrucciones a una tasa sorprendente e inesperada, 100 a 200,000 veces más rápido que el procesador estándar que enfrenta el mundo, sin reescribir programas escritos para el procesador estándar. Esta simplificación sobre la pesada

tarea de reescribir el código para funcionar en FPGAs finalmente hace que el uso de FPGAs sea accesible para programadores ordinarios. El AMPC no genera el código de tiempo de ejecución para el chip ICAT, en un ejemplo. En

5 cambio, precompila el programa y separa las instrucciones que mejor se adaptan a las RAPCs. Entonces, el AMPC genera un código para configurar cada una de una pluralidad de RAPCs en el chip ICAT (o en otro lugar, en un ejemplo, usando múltiples chips ICAT operando en paralelo), cuyos RAPCs luego

10 operan en paralelo y en canalización. Alternativamente, los RAPCs pueden reconfigurarse en tiempo real, en base a las instrucciones recibidas por el ICAT o en instrucciones históricas recibidas previamente por el chip ICAT. Por lo tanto, el chip ICAT puede aprender a través del tiempo a

15 operar a velocidades cada vez más rápidas, si el chip ICAT se usa de forma consistente para propósitos similares. Esto ocurre, naturalmente, si los RAPCs están configurados en tiempo real para resolver nuevas ecuaciones lógicas y matemáticas, mientras se mantienen las configuraciones

20 lógicas y matemáticas antiguas en su lugar en base al primero en entrar, primero en salir, último en usarse o ultimo reconfigurado. Un conjunto de RAPCs reconfigurado para un propósito específico, si se usa con frecuencia, no se reconfigurará para otro propósito hasta que no haya otros

25 RAPCs disponibles para nuevos algoritmos que requieran RAPCs, si se utiliza, por ejemplo, el último algoritmo

reconfigurado usado por última vez. De esta manera, los algoritmos más utilizados no necesitarán ser configurados, pero serán ya preconfigurados por el uso previo.

[0024] Los RAPCs, cuando están configurados por la unidad
5 de procesamiento central, funcionan sin sobrecarga, ejecutando instrucciones hasta que se completen las instrucciones matemáticas, lógicas e iterativas para las que se ha configurado el RAPC.

[0025] En un ejemplo, el chip ICAT comprende registros de
10 configuración, y el AMPC genera instrucciones para configurar los registros de configuración del chip ICAT, que configura los RAPCs para completar instrucciones particulares. Los RAPCs operan continuamente, sin supervisión adicional por parte de la unidad de procesamiento
15 central, cuando se inicializan. En un ejemplo, el AMPC recibe datos de hardware RAPC de un compilador de hardware, como un compilador de hardware Verilog o Vivado. Los archivos de hardware pueden ser generados por el compilador de hardware y pueden ser utilizados por el AMPC para generar el código
20 que escribe datos de configuración para los registros de configuración del chip ICAT (o los registros de configuración de una pluralidad de chips ICAT, en un ejemplo).

[0026] En un ejemplo, el AMPC extrae los datos de configuración para los registros de configuración de un chip
25 ICAT de un programa escrito para una arquitectura de procesamiento estándar en un lenguaje de programación de

alto nivel, tal como "C". Por ejemplo, el AMPC ignora las instrucciones generales y genera un código para los registros de configuración del chip ICAT del programa para 1) instrucciones y datos aritméticos; 2) decisiones y datos lógicos; y 3) instrucciones y destinos de bifurcación o llamada/devolución; 4) bucles iterativos, decisiones y datos; 5) rutinas y datos de configuración de DSP; y 6) etiquetas de punto de entrada de código para bucles y ramas. Por ejemplo, el AMPC utiliza estas instrucciones para configurar los registros de configuración del ICAT, configurar el DSP para completar los algoritmos matemáticos, el LDP para las decisiones lógicas y los valores para las tablas de búsqueda del LDP y el MBS para los destinos de derivación, llamada y retorno asignación de etiquetas a puntos de entrada en los diversos algoritmos de procesamiento y direcciones asignadas en el hardware ICAT. Por ejemplo, las tablas de hardware RAPC se crean para cada RAPC y contienen las tablas de configuración DSP, LDP y MBS. Por ejemplo, DSP, LDP y MBS se configuran como de uso frecuente en los RAPCs, pero cuando no se necesitan DSP o LDP, entonces un RAPC se puede reconfigurar incluso para omitir esta estructura común mediante la arquitectura AMPC e ICAT. Entonces, aunque DSP, LDP y MBS están presentes en algunos de los RAPCs, otros RAPCs pueden tener una estructura diferente, específica del código que se ejecutará en el procesador ICAT.

[0027] En un ejemplo, una arquitectura ICAT y AMPC dependen uno del otro para el hardware capaz de ser reconfigurado por el AMPC. Por ejemplo, el AMPC puede implementar una ramificación o llamada de un destino dentro
5 de la arquitectura ICAT conectando directamente resultados o datos al destino, si el RAPC de destino está cerca, o los datos se enrutan desde un DSP a un LDP o viceversa, por ejemplo, haciendo que los resultados o los datos estén inmediatamente disponibles para la ejecución de las
10 instrucciones, sin ningún gasto adicional. Alternativamente, el AMPC puede implementar una ramificación o llamada de un destino usando el MBS, y los resultados y datos son transportados en una interfaz de transmisión de alta velocidad al destino, que puede ser otro RAPC u otro destino,
15 haciendo que los datos estén disponibles para la posterior ejecución de instrucciones a través de la interfaz de transmisión de alta velocidad hasta el destino.

[0028] En un ejemplo, el AMPC conoce los recursos de RAPC, los cuales son asignados por el AMPC mientras precompila el
20 código escrito en el lenguaje de programación de alto nivel. Por lo tanto, la arquitectura ICAT puede ser configurada por el AMPC para optimizar el uso de los recursos de RAPC, por ejemplo, minimizando la longitud de la interconexión entre las instrucciones ejecutadas por la pluralidad de RAPCs.
25 Esta optimización puede completarse mediante un enfoque iterativo o un enfoque de prueba y error. En un ejemplo, el

AMPC comprende un algoritmo de trabajo en equipo que mejora las optimizaciones basadas en patrones históricos de uso de ciertas instrucciones, ya sean algoritmos matemáticos, algoritmos lógicos o una combinación de algoritmos matemáticos y lógicos, por ejemplo, minimizando el uso del MBS para la ramificación o llamada de un destino para conjuntos de instrucciones comunes. Para ver un ejemplo de una implementación de MBS, vea el ejemplo ARM MBS en segundo plano.

10 **[0029]** En un ejemplo, un RAPC está integrado en un chip con un procesador convencional para configurar el RAPC y un AMPC para compilar un código fuente de alto nivel convencional en instrucciones para que el procesador convencional configure el RAPC. El RAPC comprende un DSP, un

15 LDP y un MBS. En este ejemplo, cada DSP tiene una interfaz de configuración para programar cualquiera de una pluralidad de operaciones, por ejemplo, de número entero y matemáticas de punto flotante, tales como multiplicar, dividir, sumar, restar y otras funciones matemáticas. Un DSP puede tener

20 entradas para datos de operandos que pueden concatenarse u operarse con varias combinaciones de funciones matemáticas determinadas por los datos de configuración. En este ejemplo, cada DSP tiene un acumulador de 48 bits que se emite como datos de resultado junto con los datos de estado. Los datos

25 de estado incluyen, ejecutar, igual, mayor que y menor que, por ejemplo. En este ejemplo, cada LDP tiene una interfaz de

configuración para programar una tabla de búsqueda, un contador de bucle y un registro constante. Cada LDP tiene un "contador de bucle" para detectar cuándo se completan los algoritmos iterativos. Cada LDP tiene un registro que puede
5 contener datos constantes para la entrada a la tabla de búsqueda. Cada LDP tiene un bloque de memoria, que se puede usar para realizar funciones. Las funciones de la tabla de búsqueda pueden incluir una tabla de búsqueda a la que se puede implementar y acceder secuencialmente utilizando el
10 contador de bucles; una tabla de búsqueda que puede ser implementada y accedida por el estado del DSP, el registro constante o los datos del resultado del DSP para fines de control; y una tabla de búsqueda lógica que puede implementarse y emitir señales lógicas misceláneas para
15 fines de control, por ejemplo. El LDP puede pasar datos de resultados desde su entrada a su salida. El LDP puede tener un registro de canalización para los datos de resultado en su salida, por ejemplo. Alternativamente, el LDP puede tener dos registros de canalización con limpieza sincrónica que
20 permite obtener los resultados en su salida. Por ejemplo, el chip puede ser un chip ICAT que comprende una pluralidad de los RAPCs, cada uno de los cuales comprende un DSP, un LDP y un MBS y cada uno de ellos está configurado por el código proporcionado por el AMPC a un procesador convencional.

25 **[0030]** En un ejemplo, un AMPC comprende un compilador que tiene una arquitectura de entrada para definir el número de

una pluralidad de RAPCs y las ubicaciones de la pluralidad de RAPCs. El AMPC filtra el código fuente de alto nivel e identifica algoritmos matemáticos y lógicos que pueden optimizarse mediante la configuración de uno o más de la pluralidad de RAPCs. Por ejemplo, si se identifica un algoritmo matemático o lógico de procesamiento de video, el AMPC configura el DSP, LDP y MBS de uno o más de los RAPCs para realizar el procesamiento de video, el algoritmo matemático y/o lógico. Por ejemplo, AMPC crea código de máquina a partir de un código fuente de lenguaje "C" para el funcionamiento de un procesador convencional, como un procesador ARM, y el procesador ARM configura cada una de las porciones DSP, LDP y MBS de cada uno de los RAPC que se utilizará en el procesamiento de la entrada de datos al procesador y la salida de datos del procesador.

[0031] Para sistemas fuera del procesador, un procesador ICAT parecerá ser un procesador convencional inusualmente rápido. Dentro del procesador, los DSP, LDP y MBS de los RAPC procesarán datos a una velocidad decenas, cientos, incluso miles de veces más rápido que un procesador convencional de un solo núcleo. Para cada RPAC, el DSP realizará su operación en un primer reloj, el LDP probará el resultado y emitirá una decisión de control y los resultados en un segundo reloj, y el MBS enrutará los datos del resultado a uno de dos destinos según el control datos en un tercer reloj. Por lo tanto, cada RAPC tendrá una latencia de

3 relojes desde DSP a MBS. Para la transmisión de datos, una vez iniciado, el MBS puede generar datos en cada reloj posterior después del período de latencia.

[0032] En un ejemplo, un sistema para configurar un procesador reconfigurable comprende un procesador no reconfigurable, una pluralidad de núcleos reconfigurables, y un compilador canalizado de correlación algorítmica capaz de aceptar código escrito en un lenguaje de programación de alto nivel para el procesador no reconfigurable, en donde el compilador identifica el código escrito en el lenguaje de programación de alto nivel que podría beneficiarse de la canalización disponible en uno o más de la pluralidad de núcleos reconfigurables y el código de salidas para el procesador no reconfigurable para configurar uno o más de la pluralidad de procesadores no reconfigurables .

[0033] En un ejemplo, un procesador comprende un núcleo de procesador no reconfigurable y una pluralidad de núcleos canalizados algorítmicos reutilizables acoplados al núcleo del procesador no reconfigurable de modo que el núcleo del procesador no reconfigurable es capaz de configurar y reconfigurar cada uno de la pluralidad núcleos canalizados algorítmicos reutilizables como resultado de las instrucciones recibidas de un compilador canalizado correlación algorítmica. Por ejemplo, el procesador está contenido en un solo chip. En un ejemplo, cada núcleo canalizado algorítmico reutilizable comprende un DSP, un LDP

y un MBS, y el DSP se canaliza al LDP, y el LDP se canaliza al MBS, de modo que el procesador no reconfigurable no controle ninguno de los procesamientos que ocurren dentro de cada núcleo canalizado algorítmico reutilizable.

5 **[0034]** DEFINICIONES. Un compilador canalizado de correlación algorítmica o AMPC es un compilador capaz de aceptar código escrito en un lenguaje de programación de alto nivel para un procesador convencional no reconfigurable, donde el AMPC identifica el código escrito

10 en el lenguaje de programación de alto nivel que podría beneficiarse de la canalización disponible en un núcleo o procesador reconfigurable, como RAPC o arreglo de puertas programables archivadas, y código de salidas para un procesador no reconfigurable, que indica al procesador no

15 reconfigurable que configure el núcleo o procesador reconfigurable, antes de proporcionar instrucciones para usar el núcleo o procesador reconfigurable. Un núcleo (o computadora) canalizado algorítmico reutilizable (o reconfigurable) o RAPC se define como un núcleo de

20 procesamiento reconfigurable con una estructura canalizada que comprende un DSP que incluye una interfaz de configuración para programar cualquiera de una pluralidad de operaciones, tales como número entero y matemáticas de punto

25 flotante, con cuatro entradas para datos de operandos que pueden concatenarse u operarse con varias combinaciones de funciones matemáticas según lo determinado por los datos de

configuración, y un acumulador de 48 bits que se emite como datos de resultado junto con los datos de estado; un LDP que tiene una interfaz de configuración para programar una tabla de búsqueda, un contador de bucles, un registro constante y
5 un bloque de memoria que se puede usar para realizar funciones; y un MBS. Un MBS se define como un circuito programable reconfigurable que enruta los datos y los resultados de un RAPC a otro RAPC y desde/hacia un controlador de entrada/salida, y/o interrumpe los
10 generadores, según sea necesario, para completar un algoritmo, sin ninguna intervención adicional de un procesador central o periférico durante el procesamiento del algoritmo.

BREVE DESCRIPCION DE LOS DIBUJOS

15 **[0035]** Los siguientes dibujos son ejemplos ilustrativos y no limitan aún más las reivindicaciones que eventualmente puedan emitirse.

[0036] La figura 1A ilustra un diagrama de flujo de la técnica anterior para un compilador convencional.

20 **[0037]** La Figura 1B ilustra un procesador de la técnica anterior para una computadora convencional.

[0038] La Figura 2 ilustra un diagrama de bloques de la patente de EE.UU. N° 5.684.980.

[0039] La Figura 3 es un diagrama de flujo que ilustra un
25 ejemplo de un compilador de AMPC para su comparación con el diagrama de flujo de la Figura 1A.

[0040] La Figura 4 es un ejemplo de una arquitectura ICAT.

[0041] La Figura 5 muestra un diagrama de flujo de un ejemplo de cómo un programador puede usar un AMPC.

[0042] La Figura 6 es un ejemplo esquemático de una
5 computadora canalizada algorítmicamente reutilizable.

[0043] La Figura 7 muestra una ilustración esquemática de un diagrama de una configuración de hardware resultante de una compilación del Ejemplo de Código 1 con un compilador AMPC.

10 **[0044]** La Figura 8 ilustra un beneficio dramático de la potencia de procesamiento bruta del ejemplo de la Figura 7 mediante compresión de datos sin pérdida en tiempo real en un dispositivo electrónico de consumo.

[0045] Cuando se usan los mismos caracteres de
15 referencia, estas etiquetas se refieren a partes similares en los ejemplos ilustrados en los dibujos.

DESCRIPCION DETALLADA

[0046] Por ejemplo, una arquitectura ICAT imita cualquier arquitectura de unidad de microprocesador estándar. Su
20 arquitectura aprovecha la canalización y una densidad de puertas mucho más rica en un circuito integrado diseñado para ser configurado por un cliente o diseñador después de la fabricación, como uno o más arreglos de puertas programables de campo (FPGAs) para lograr una ventaja de
25 100: 1 en MIPS cuando se realiza una comparación 1: 1 con una única arquitectura de microprocesador estándar con la

misma velocidad de reloj. Los FPGAs contienen un arreglo de bloques lógicos programables y una jerarquía de interconexiones reconfigurables que permiten que los bloques se "conecten entre sí", como muchas puertas lógicas que
5 pueden interconectarse en diferentes configuraciones. Los bloques lógicos se pueden configurar para realizar funciones combinacionales complejas, o simplemente puertas lógicas simples como AND y XOR. En la mayoría de los FPGAs, los bloques lógicos también incluyen elementos de memoria, que
10 pueden ser flip-flops simples o bloques de memoria más completos.

[0047] El gran salto en el rendimiento permite que el procesador se utilice para aplicaciones intensivas de datos, como visión de máquina, procesamiento de video,
15 procesamiento de audio, sistemas de control robótico, sistemas de control multi-eje, comunicaciones móviles, realidad virtual, inteligencia artificial, transmisión en vivo, monitoreo biométrico, internet de las cosas, supercomputación, computación cuántica, sistemas de control
20 aeroespacial, simulación y modelado de sistemas complejos, y aplicaciones de procesamiento de señales, por ejemplo.

[0048] En un ejemplo, se utiliza menos potencia para el procesamiento computacionalmente intensivo de algoritmos. Por ejemplo, la arquitectura ICAT proporciona una reducción
25 de 100 a 1 en el uso de energía para el mismo cálculo implementado en una unidad de microprocesamiento estándar,

más preferiblemente una ventaja de 1000: 1, reduciendo el calor y el consumo de energía.

[0049] En un ejemplo, el ICAT puede ejecutarse en una configuración de tantos procesadores paralelos como sea necesario para una aplicación, aumentando el rendimiento aún más en comparación con los microprocesadores estándar. Por ejemplo, una pluralidad de arquitecturas de procesador se puede ejecutar simultáneamente. Por ejemplo, el código heredado puede ejecutarse en una máquina virtual compatible con el código heredado, mientras que una nueva máquina virtual ejecuta código escrito específicamente para la nueva arquitectura. En un ejemplo, esto reduce la necesidad de realizar extensas pruebas de regresión, como las que serían necesarias para adaptar el código heredado a la nueva arquitectura del sistema.

[0050] En una aplicación, la velocidad y la capacidad de expansión de la arquitectura ICAT se aplica a los sistemas heredados que no pueden procesar el volumen de datos necesarios para la velocidad bruta y la capacidad de expansión para los clientes cuyo código y/o hardware se ha topado con limitaciones.

[0051] En un ejemplo, la reconfiguración se compila en o antes del encendido, simplificando enormemente la planificación con poco impacto en el rendimiento del producto final. Por ejemplo, un FPGA es un huésped del hardware para esta arquitectura. Se pueden agregar millones de

instrucciones por segundo (MIPS) fácilmente, sin reescribir el código existente. El código existente puede ejecutarse casi sin modificaciones, a excepción de la recompilación del código existente. Por ejemplo, los algoritmos que requieren

5 el procesamiento paralelo de una gran cantidad de entradas comunes son candidatos ideales para esta arquitectura ICAT.

[0052] En un ejemplo, procesadores antiguos y nuevos se ejecutan en paralelo. El código existente puede recompilarse y ejecutarse casi intacto, con un mínimo de pruebas de

10 regresión para garantizar que no se hayan producido cambios. Habrá excepciones cuando el tiempo afecte las operaciones de la arquitectura y donde los periféricos de hardware estén alterados. Por ejemplo, una arquitectura ICAT puede usarse para aumentar la velocidad de cálculo en bruto, y la

15 aceleración del código puede implementarse mediante la conversión de hardware cuando sea necesario.

[0053] En un ejemplo, la arquitectura ICAT comprende un precompilador frontal que capta cualquier posible problema de incompatibilidad de código. Este precompilador frontal

20 resuelve automáticamente estos posibles problemas de incompatibilidad de código. Por ejemplo, la arquitectura ICAT puede emular una variedad de arquitecturas de procesador familiares para diferentes desarrolladores. Por ejemplo, la arquitectura ICAT puede emular más de un procesador,

25 permitiendo codificar un proyecto para una pluralidad de procesadores favorecidos de los desarrolladores y ejecutar

el código en una pluralidad de procesadores virtuales diferentes al mismo tiempo. En un ejemplo, una pluralidad de procesadores diferentes ejecutaría conjuntos de códigos diferentes en un entorno de multiprocesamiento, y los desarrolladores de programa compilarían código para uno de la pluralidad de dominios compatibles con el código.

[0054] En un ejemplo, el precompilador es un compilador canalizado de correlación algorítmica, el cual genera el código de configuración de hardware necesario para varios algoritmos de procesamiento. El firmware para configurar la arquitectura ICAT se puede generar a partir de ecuaciones lógicas y matemáticas para una pluralidad de tareas de procesamiento. Por ejemplo, una pluralidad de procesadores puede configurarse en un arreglo de matrices para ejecutar una mezcla de tareas de bajo y alto rendimiento.

[0055] La arquitectura ICAT incluye código de procesamiento desarrollado utilizando un lenguaje de nivel superior, porque la arquitectura ICAT proporciona una ventaja de velocidad cruda que supera cualquier ventaja de velocidad obtenida por la programación en lenguaje de máquina aplicable solo a un entorno de multiproceso específico, reduciendo sustancialmente el tiempo para completar un proyecto de desarrollo.

[0056] La arquitectura ICAT incluye un compilador o precompilador, que comprueba el código heredado para comandos específicos del hardware, que está optimizado para

usar con un lenguaje de programación de alto nivel, como C o C ++. Una comparación de la Figura 1 y la Figura 3 ilustra los pasos adicionales incluidos en un compilador canalizado de correlación algorítmica (AMPC), por ejemplo.

5 **[0057]** En un ejemplo, la arquitectura ICAT proporciona un conjunto de periféricos estándar de multiprocesamiento/multitarea, con coordinación incorporada. Se puede adoptar un sistema operativo en tiempo real (RTOS). Por ejemplo, un sistema operativo multitarea en

10 tiempo real se incorpora a la arquitectura ICAT. Por ejemplo, Micro-Controller Operating Systems (MicroC/OS) es un sistema operativo en tiempo real diseñado por el desarrollador de software integrado, Jean J. Labrosse en 1991. Es un sistema operativo preventivo en tiempo real basado en prioridad para

15 microprocesadores, escrito principalmente en el lenguaje de programación C, un lenguaje de programación de nivel superior. La velocidad bruta de la arquitectura ICAT permite el uso de dicho RTOS, por ejemplo. MicroC/OS permite la definición de varias funciones en el lenguaje C, cada una de

20 las cuales se puede ejecutar como un hilo o tarea independiente. Cada tarea se ejecuta con una prioridad diferente, y cada tarea piensa que posee un procesador virtual de la arquitectura ICAT. Las tareas de menor prioridad pueden ser reemplazadas por tareas de mayor

25 prioridad en cualquier momento. Las tareas de mayor prioridad pueden usar servicios del sistema operativo, como un retraso

o evento, para permitir la ejecución de tareas de menor prioridad. Hay servicios del sistema operativo proporcionados para la administración de tareas, la comunicación entre tareas, la administración de memoria y para sincronizar MicroC/OS. MicroC/OS es de código abierto y se puede adaptar a varias arquitecturas de procesador diferentes.

[0058] El software de diseño de PCB y las herramientas de ingeniería son proporcionados para la arquitectura ICAT con el fin de permitir que los diseños existentes se conviertan a la arquitectura ICAT.

[0059] En un ejemplo, una arquitectura en línea se logra utilizando el código estándar Verilog o VHDL. Por ejemplo, se puede proporcionar una memoria caché de instrucciones de 1024 palabras, un caché de datos y arquitecturas de caché de memoria de varios niveles en la arquitectura ICAT. La canalización de la arquitectura ICAT puede incluir un algoritmo de aprendizaje que detecta de qué manera tiende a ocurrir la ramificación en el proceso de decisión, haciendo que esa ruta sea la ruta predeterminada en el futuro a través del algoritmo de aprendizaje. En otro ejemplo, el código de interrupción está aislado, y un controlador de interrupción está dedicado a entradas específicas, con una ubicación de código privado. En un ejemplo, la arquitectura ICAT incluye un depurador multiprocesador. Por ejemplo, el código existente puede ser procesado por un depurador de

preprocesamiento para garantizar que el código existente esté bien particionado, de modo que las funciones estén separadas. Luego se puede ejecutar un único depurador en cada subproceso independiente de una operación.

5 **[0060]** Por ejemplo, se puede proporcionar un núcleo canalizado algorítmico reconfigurable (RAPC) en un paquete de chips de 2 pulgadas que proporciona MIPS y Mega FLOPS equivalentes a más de 1000 microprocesadores Intel i7, preferiblemente más de 10,000 microprocesadores Intel i7.

10 **[0061]** En un ejemplo, la arquitectura ICAT es compatible con las herramientas de depuración existentes. En otro ejemplo, la arquitectura ICAT se implementa para ejecutar código heredado existente que no contiene comunicaciones entre procesadores. El hardware específico de ICAT está
15 unificado como un único bloque bien depurado común a todos los códigos heredados. Por ejemplo, los periféricos que imitan exactamente las funciones principales de las unidades comunes de multiprocesamiento se clonan para la arquitectura ICAT. Por ejemplo, los periféricos superconfigurados
20 permiten arreglos de hardware fácilmente organizados por los clientes.

[0062] En un ejemplo, el compilador o precompilador de arquitecturas ICAT detecta ciclos de temporización de códigos de bajo nivel que cuentan ciclos de reloj, retrasos
25 que permiten la recuperación de instrucciones y otros códigos de temporización incompatibles, y los marca para su

reparación o reemplazo, manual o automáticamente, con compatibilidad programación de alto nivel proporcionada dentro de la arquitectura ICAT.

[0063] En un ejemplo, la arquitectura ICAT proporciona una ventaja MIPS 4: 1 sobre las arquitecturas tradicionales. En otro ejemplo, la ventaja es al menos 100: 1.

[0064] En un ejemplo, la arquitectura ICAT comprende un compilador canalizado de correlación algorítmica (AMPC), que es un compilador que acepta algoritmos de procesamiento en formatos de código fuente estándar. El AMPC genera firmware para un sistema de procesamiento convencional operable con la arquitectura ICAT. El compilador genera instrucciones que configuran el hardware de ICAT, de modo que la arquitectura procesa algoritmos con un rendimiento mejorado en comparación con los microprocesadores tradicionales que no son reconfigurables mediante AMPC. Específicamente, el AMPC utiliza la canalización para optimizar el rendimiento del procesador para aplicaciones que requieren un procesamiento computacional algorítmico intensivo. Por ejemplo, este firmware se puede ejecutar en un sistema de procesamiento convencional para configurar arquitecturas de hardware de ICAT(s) que procesen algoritmos con un rendimiento óptimo.

[0065] En un ejemplo, el AMPC proporciona un compilador que compila el código fuente del compilador convencional capaz de generar código para operar el hardware ICAT, configurando los recursos del procesador de la arquitectura

ICAT para procesar directamente los algoritmos. Por ejemplo, el AMPC utiliza código fuente que es compatible con compiladores convencionales, como C, C #, C ++, Matlab u otros compiladores convencionales.

5 **[0066]** En un ejemplo, el firmware generado por el AMPC se ejecuta en un sistema de procesamiento principal de la arquitectura ICAT. Por ejemplo, el sistema de procesamiento principal es un procesador convencional en el mismo chip que el resto de la arquitectura ICAT y funciona a la perfección

10 con la arquitectura ICAT. En este ejemplo, el AMPC acepta código escrito en lenguajes de programación de alto nivel para el código fuente, como C, C #, C ++, y el firmware de salidas AMPC para la arquitectura ICAT que se ejecuta en el sistema de procesamiento principal. Esto simplifica la

15 codificación para el funcionamiento de la arquitectura ICAT al permitir que el firmware de la arquitectura ICAT se programe en un lenguaje de programación de nivel superior familiar para el desarrollador. La velocidad bruta de la arquitectura ICAT elimina la penalización y reduce la

20 necesidad de programar el código del nivel de la máquina para optimizar la velocidad. En cambio, el lenguaje de programación de nivel superior optimiza el firmware para optimizar el rendimiento en función de los algoritmos que se deben resolver para una aplicación en particular. Por

25 ejemplo, la arquitectura ICAT es reconfigurable para permitir un rendimiento óptimo, en al menos una máquina

virtual definida en el firmware, para sistemas de visión robótica, por ejemplo.

[0067] A diferencia de los microprocesadores tradicionales, en un ejemplo, el AMPC de la arquitectura ICAT puede compilar sintaxis de software, como un proceso if-then-else, en firmware que reconfigura el hardware de la arquitectura ICAT para ejecutar el proceso de manera óptima en menos ciclos de reloj, usando la canalización, por ejemplo. Al ejecutar el firmware, se configura la arquitectura ICAT. En contraste, los compiladores convencionales crean el firmware que usan todos los procesadores convencionales, pero los procesadores convencionales no se reconfiguran con el firmware. El AMPC crea firmware para la arquitectura ICAT, configurando la arquitectura ICAT para un funcionamiento óptimo en una aplicación particular, por ejemplo. En un ejemplo, el AMPC selecciona y estructura la configuración del hardware ICAT utilizando los algoritmos como estructura de entrada para el hardware del procesador de la arquitectura ICAT.

[0068] Por ejemplo, la arquitectura de hardware de la arquitectura ICAT se optimiza mediante el AMPC para procesar el rendimiento de velocidad para una aplicación en particular, cuando se configura mediante el firmware AMPC generado. El AMPC puede reconfigurar el hardware de la arquitectura ICAT, donde un compilador convencional no puede reconfigurar el ICAT o el hardware de ningún microprocesador.

[0069] Un compilador de sistema estándar no puede cambiar la arquitectura del hardware en sistemas de procesador convencionales. Sin embargo, en un ejemplo, AMPC genera un firmware que configura los procesadores de arquitecturas

5 ICAT para realizar directamente el procesamiento segmentado y el enrutamiento de los datos en función de los resultados previos en el hardware. Por ejemplo, la entrada de la declaración lógica if-then-else en el AMPC estructuraría el hardware de la arquitectura ICAT para enrutar los resultados

10 de los datos al próximo ICAT. En este ejemplo, el AMPC genera configuraciones de hardware que eliminan la sobrecarga de los sistemas de procesamiento convencionales, como la búsqueda de códigos, la carga de datos, el almacenamiento de datos, la ramificación y las subrutinas para la misma lógica

15 if-then-else.

[0070] La Figura 4 ilustra un ejemplo de una arquitectura ICAT. En un ejemplo, un compilador convencional, como Visual Studio, se puede usar para generar un programa de configuración de ICAT que se ejecuta en el sistema de

20 procesamiento principal 101. Esto proporciona un método para configurar y reconfigurar grupos de hardware reprogramables que son reconfigurables para ejecutar y procesar varios algoritmos de procesamiento de tipo en un chip. Un sistema de procesamiento convencional (por ejemplo,

25 microprocesadores Intel, ARM, IBM, AMD) no se puede reconfigurar para ejecutar varios algoritmos, porque solo el

software, no el hardware, puede cambiar en un sistema de procesamiento convencional. Mediante el uso de una arquitectura ICAT, se elimina toda la sobrecarga de instrucción de código de búsqueda y ejecución de un sistema de procesamiento convencional. La arquitectura de ICAT de la Figura 4 proporciona un hardware configurable para realizar un procesamiento eficiente de datos que utiliza un conjunto de recursos de procesadores paralelos implementados en un dispositivo en un dispositivo en un chip (SOC) dispositivo

10 100.

[0071] Por ejemplo, un grupo de procesadores matemáticos 107, seguido por procesadores lógicos 108 y enrutamiento de matriz configurable 109 implementa un conjunto de recursos de procesamiento paralelos 102. Esta arquitectura es capaz de recursos de procesamiento de canalización para optimizar el rendimiento de procesamiento para aplicaciones particulares. En un ejemplo, el grupo de procesadores 102 realiza múltiples tareas de procesamiento, independientemente del procesador principal 101, sin recibir más instrucciones del procesador principal. Cada ICAT puede configurarse para procesar un algoritmo completo como un sistema de procesador independiente. Por lo tanto, un ICAT puede considerarse un sistema dentro de sí mismo, que no requiere sobrecarga para completar el procesamiento de un

15 20 25

algoritmo, una vez configurado para realizar el algoritmo. Por ejemplo, un ICAT puede configurarse para realizar un

conjunto de instrucciones if-then-else y puede reconfigurarse, más tarde, para realizar un conjunto de instrucciones completamente diferente, como una transformada rápida de Fourier u otra solución de algoritmo matemático.

5 **[0072]** Al reducir los ciclos de actividad innecesarios, la arquitectura ICAT reduce el consumo de energía, genera menos calor y aumenta la velocidad de procesamiento de los datos, en comparación con un procesador convencional. Los recursos de ICAT 102 están inactivos hasta que se configuran,
10 cuando los datos están listos para procesarse en sus entradas. Todos los procesadores se mantienen inactivos cuando no son necesarios, reduciendo el calor generado por cualquier sobrecarga innecesaria. Cada procesador en el conjunto de recursos de ICAT tiene menos sobrecarga que los
15 procesadores convencionales, porque el ICAT no busca y ejecuta el código. En cambio, el hardware se configura para realizar una operación específica y solo está activo cuando se proporcionan datos que deben procesarse con el algoritmo configurado proporcionado por la arquitectura ICAT. En un
20 ejemplo, un solo procesador ICAT usa un conjunto de procesadores matemáticos 107, procesadores lógicos 108 y salida dirigida por el enrutamiento de matriz configurable 109.

[0073] Este mismo procesador ICAT se puede utilizar para
25 una tarea de procesamiento simple, como un if-then-else, o para un algoritmo complejo muy avanzado, como un algoritmo

utilizado en el reconocimiento facial. Utilizando una pluralidad de grupos o conjuntos de recursos ICAT 102, un conjunto de procesadores matemáticos 107, procesadores lógicos 108 y salida dirigida por el enrutamiento de matriz configurable 109, la arquitectura ICAT puede usarse para procesar tareas que requieren una pluralidad de cálculos en una arquitectura canalizada, como detección de movimiento, forma o identidad, por ejemplo.

[0074] En un ejemplo, el algoritmo controla la estructura del bus de interconexión de los procesadores ICAT, y la arquitectura ICAT procesa flujos de datos de entrada desde dispositivos de salida 112, tales como video, sensores o datos de un paso de proceso previo. Por ejemplo, los resultados previos pueden transmitirse desde almacenamientos intermedios de memoria de datos, datos de entrada en vivo o cualquier información de otros pasos procesados 110, 111. Los resultados del procesamiento pueden enviarse directamente a dispositivos 113, tales como salida de control o salida de video, por ejemplo.

[0075] Un programador puede utilizar el AMPC para configurar una pluralidad de RAPCs como se ilustra en el ejemplo de la Figura 5. Alternativamente, el uso del AMPC puede ser automatizado y controlado a bordo por un sistema en un chip, por ejemplo. La Figura 5 ilustra un diagrama de flujo de 6 pasos para un programador, que inicialmente inserta un código original de lenguaje de programación de

alto nivel en el primer compilador (el AMPC se refiere a un ASML). El precompilador de ASML extrae el código de la fuente original en el paso 2, que se produce automáticamente. Luego, el precompilador genera un nuevo código fuente para un
5 segundo compilador. Este paso puede ser hecho automáticamente o como un paso separado por el programador, después de que el programador esté satisfecho con la depuración y optimización de la nueva fuente. Este segundo compilador compila una compilación de firmware para la
10 arquitectura ICAT. Luego, el firmware se carga en la arquitectura ICAT y el firmware configura los RAPCs de la arquitectura ICAT. El programador puede cargar este firmware en la arquitectura ICAT después de que el programador esté satisfecho de que el firmware está depurado y optimizado,
15 por ejemplo.

[0076] Alternativamente, cada uno de los pasos puede ser automatizado y puede ocurrir sin intervención humana, excepto para cargar el código fuente original en la arquitectura ICAT. Al combinar un procesador convencional
20 con una pluralidad de RAPCs y un AMPC, todo el proceso puede ser automatizado, de modo que el procesador convencional ejecuta el AMPC para recompilar el código fuente original y generar el firmware que utiliza el procesador convencional para configurar los RAPCs, basado en las instrucciones
25 contenidas en el código fuente original.

[0077] Un conjunto de recursos de ICAT puede contener tres tipos de módulos de procesador, por ejemplo, como módulos matemáticos, módulos lógicos y módulos de enrutamiento de resultados. Los módulos matemáticos realizan funciones matemáticas. Los módulos lógicos realizan funciones lógicas. Los módulos de enrutamiento de resultados realizan funciones de ramificación y enrutamiento de datos. Por ejemplo, en la Figura 6, se ilustra esquemáticamente un Computador Canalizado Algorítmico Reutilizable (RAPC). Un bus de configuración 109 se establece mediante la configuración de los registros de configuración de la arquitectura ICAT por parte del AMPC. Los operandos son dirigidos a las ubicaciones de memoria A, B, C y D en un procesador de señal digital (DSP) 110. El DSP está configurado para ejecutar un algoritmo matemático. Los resultados del algoritmo se dirigen a un procesador de decisión lógica (LDP) 111. El LDP ejecuta instrucciones lógicas. Los resultados de las instrucciones lógicas se entregan al siguiente RAPC, directamente o a través del interruptor de bus de matriz (MBS). El MBS dirige los resultados al siguiente RAPC o controla las entradas y salidas e interrumpe la entrega de los resultados en una interfaz de transmisión de alta velocidad.

[0078] Los recursos de hardware se pueden configurar en los sistemas de coprocesador ICAT que están interconectados en una estructura canalizada para un rendimiento óptimo. En

un ejemplo, un método para diseñar agrupaciones reprogramables de recursos de hardware, que son reconfigurables, ejecuta y procesa una pluralidad de algoritmos de procesamiento en un chip. Los recursos de hardware para configurar procesadores ICAT se pueden diseñar en el chip, y los recursos de hardware en el chip se pueden reconfigurar a través de AMPC. La arquitectura de un sistema de procesamiento ICAT se configura a partir del código fuente para algoritmos de procesamiento, por ejemplo. Por lo tanto, el código generado para un procesador convencional puede ejecutarse mucho más eficientemente en una arquitectura ICAT, porque el hardware de los procesadores ICAT está configurado por el código fuente para ejecutar algoritmos independientemente del procesador que usa AMPC, por ejemplo. Por lo tanto, la arquitectura ICAT es capaz de configurar la arquitectura de hardware ICAT a partir del código fuente creado para un microprocesador convencional, que no se conoce en la técnica. En un ejemplo, se crea un conjunto de recursos de hardware que son configurables y reconfigurables en estructuras de matriz algorítmica por un procesador, y el conjunto de recursos de hardware procesa en realidad una pluralidad de algoritmos de procesamiento en un chip. En un ejemplo, los recursos de hardware procesan los datos a través de una pluralidad de comandos independientemente de otros procesadores que usan la canalización.

[0079] En un ejemplo, la arquitectura ICAT y el compilador de canalización de correlación algorítmica se combinan para lograr resultados desconocidos en la técnica, logrando tanto la velocidad como la eficiencia de los cálculos. Por ejemplo, un AMPC configura recursos de hardware para ejecutar una pluralidad de algoritmos de procesamiento. AMPC genera el firmware de configuración utilizado para configurar los algoritmos de procesamiento del conjunto de recursos de ICAT en un chip ICAT. Esto proporciona a un programador una herramienta que acepta el código fuente de la aplicación existente, diseñado para un procesador convencional y un nuevo código fuente diseñado para emparejar y asignar recursos de hardware ICAT para crear algoritmos de procesamiento de hardware individuales dentro de la arquitectura ICAT. AMPC genera el firmware que ejecuta el procesador principal para configurar el hardware ICAT para realizar una pluralidad de algoritmos independientes del procesador principal, durante el funcionamiento del SOC para un propósito particular.

[0080] Los procesadores convencionales usan una arquitectura similar que comprende la memoria de programa, el hardware de búsqueda y ejecución que se utiliza para la ejecución paso a paso de las instrucciones del programa; memoria de datos que es necesaria para el almacenamiento de datos a granel (cúmulo) y estructuras de pila de programas; y ciclos de búsqueda y ejecución de instrucciones, gestión

de la pila de programas y gestión del almacenamiento de almacenamiento dinámico de datos, que crean una sobrecarga considerable en una arquitectura de procesador convencional.

[0081] Por el contrario, en un ejemplo, una arquitectura

5 ICAT elimina casi todos los gastos generales de los sistemas de procesadores convencionales. El conjunto de hardware de ICAT está configurado por AMPC y se utiliza para procesar algoritmos utilizando la arquitectura de coprocesador ICAT con estructuras de datos de transmisión en cascada. Por lo

10 tanto, en un ejemplo, un método que utiliza la arquitectura ICAT comprende AMPC que accede a tablas de compiladores de hardware ICAT que definen los recursos disponibles en el chip; un lenguaje de diseño de hardware, como Verilog, se utiliza para compilar el conjunto de hardware ICAT 102, para

15 un procesador determinado; la compilación de hardware genera tablas que definen la estructura de los agrupamientos de recursos de ICAT dentro del chip; el AMPC usa estas tablas de datos generadas por el compilador de hardware para determinar las ubicaciones y cantidades de recursos de ICAT

20 en el chip; AMPC asigna recursos de hardware, configura operaciones matemáticas y lógicas, y crea interconexiones para los diversos algoritmos, en donde la sintaxis de entrada de fuente para el AMPC puede estar compuesta por sintaxis C # o sintaxis matemática estándar, como Matlab; el AMPC

25 configura una estructura canalizada para cada algoritmo del conjunto de recursos de hardware de ICAT que están

disponibles 103 ... 111; y estas estructuras canalizadas forman coprocesadores de ICAT para cada algoritmo, por ejemplo. Por ejemplo, el código de salidas de AMPC que se ejecuta en el sistema de procesamiento principal 101 que
5 configura los registros de control 103, 104, 105, 106 de los recursos que ejecutan algoritmos en los coprocesadores paralelos 102 de ICAT.

[0082] Una estructura de sistema de coprocesador puede configurarse a partir de un conjunto de recursos de ICAT
10 102, que responden a la entrada de un procesador principal 101, por ejemplo. Alternativamente, un grupo de recursos de ICAT 102 puede generar interrupciones y datos de salida al procesador principal 101 o dispositivos de entrada/salida del procesador principal 101, si la arquitectura del
15 procesador principal incluye dispositivos de entrada/salida separados del procesador principal. En un ejemplo, un grupo de recursos de ICAT 102 puede configurarse mediante un procesador convencional 101, luego los recursos de ICAT 102 se ejecutan solos hasta que se reconfiguran.

20 **[0083]** Los procesadores de la arquitectura ICAT procesarán continuamente flujos de datos en paralelo, por sí solos, una vez que los procesadores ICAT estén configurados por el firmware. En contraste, un sistema convencional requiere ir interminablemente a la memoria y obtener
25 instrucciones para determinar el flujo del proceso en cada paso de procesamiento. El AMPC puede asignar un grupo de

recursos de hardware, tales como lógica matemática y enrutamiento, por ejemplo, a una estructura de procesador ICAT particular de la arquitectura ICAT para ejecutar pasos de procesamiento para el procesamiento de un algoritmo particular, por ejemplo. Ningún compilador convencional selecciona y configura las estructuras de hardware de un microprocesador. Por ejemplo, cuando AMPC construye la estructura de hardware de la arquitectura ICAT, puede configurar los recursos de hardware para una arquitectura ICAT en una arquitectura enrutada que agiliza el rendimiento del procesamiento. Un compilador convencional no puede hacer esto.

[0084] En el ejemplo de la figura 4, los registros de control ICAT 104 son un conjunto de registros para controlar las funciones de procesamiento. Por ejemplo, un registro de modo de entrada del procesador de señal digital (DSP) puede incluir palabras de entrada dividida, control de preadjudicación, selección de banco de registro de entrada y otras funciones de entrada DSP, DSP ALU modo registro puede controlar sumar, restar, multiplicar, dividir, desplazar a la derecha, desplazar hacia la izquierda, rotar, y, o, xor, nor nand y otros procesos lógicos, y DSP Multiplexor Selects puede controlar los cambios y las selecciones de entrada. El DSP puede utilizar un DSP48E1 para cada ICAT. Por ejemplo, los dispositivos DSP48E1 se pueden proporcionar en una serie Xilinx 7 de arreglos de puertas programables de campo. Por

ejemplo, una memoria ICAT y operaciones lógicas 105 se pueden usar para controlar las operaciones de la memoria y la lógica de la memoria.

[0085] En un ejemplo, un algoritmo de detección de movimiento está escrito en el lenguaje C para su uso en una computadora de propósito general.

[0086] Ejemplo de código 1: algoritmo de detección de movimiento escrito en el lenguaje C (un lenguaje de programación de alto nivel)

```
10 int noise_threshold = 3;
    int live_videopixel = 0;
    int black_video_pixel = 0;
    boolean motion_detected = false;
    int live_red_pixel = 0;
15 int live_green_pixel = 0;
    int live_blue_pixel = 0;
    int frame_delayed_pixel;
    int frame_delayed_red_pixel = 0;
    int frame_delayed_green_pixel = 0;
20 int frame_delayed_blue_pixel = 0;
    int red_mask = 255; // 0000FF hex, 000000000000000011111111
    binary, extracts bits 0 thru 7 int green_mask = 65,2280; //
    0000FF hex, 000000001111111100000000 binary, extracts bits
    8 thru 15
25 int green_divisor = 256
```

```

int    blue_mask    =    16,711,680;    //    FF0000    hex,
11111111000000000000000000 binary, extracts bits 23 thru 16
int blue_divisor = 65,536

// Procedure to extract red, green, and blue pixels from
5  live _video_pixel

void extract live_red_green_blue (int live_video_pixel)
{
    live_red_pixel = (live video_pixel); // extract red pixel
    from live video
10  live_red_pixel = live_red_pixel & red_mask
    live_red_pixel = (live_red_pixel/red divisor)
    live green pixel = (live_video_pixel) ;// extract green pixel
    from live video live_green_pixel = live_ green_pixel &
    green_mask
15  live_green_pixel = (live_green_pixel/green_divisor)
    live blue pixel = (live_video_pixel) ;// extract blue pixel
    from live video
    live_blue_pixel = live_ blue_pixel & blue_mask
    live_blue_pixel = (live blue_pixel / blue_divisor)
20  }

//Procedure to extract red, green, and blue pixels from
delayed_frame_video_pixel

void extract delayed_red_green_blue (int frame delayed
video_pixel) ;
25  {

```

```

frame_delayed_red_pixel = (live_video_pixel) ;//extract red
pixel;

frame_delayed_red_pixel    =    (frame_delayed_red_pixel    &
red_mask) ;

5  frame_delayed_red_pixel    =    (frame_delayed_red_pixel/red
divisor);

frame_delayed_green_pixel = (live video_pixel) ;// extract
green pixel

frame_delayed_green_pixel  =  (frame_delayed_green_pixel  &
10 green_mask) ;

frame_delayedgreen_pixel                                     =
(frame_delayed_green_pixel/green_divisor);

frame_delayed_blue_pixel = (frame_delayed_video_pixel) ;///
extract blue pixel

15 frame_delayed_blue_pixel = (frame delayed_ blue_pixel & blue
mask)

frame_delayed_blue_pixel  =  (frame_delayed_blue_pixel/blue
divisor)

}

20 //Procedure for motion detection algorithm

boolean motion_detected detect motion () ;

{

motion detect = false;//get ready to detect motion

result_red_pixel          =          (frame_delayed_red_pixel          -
25 live_red_pixel) ; // subtract red pixels

```

```

if (result_red_pixel > noise_threshold) // test if red result
is greater than noise threshold
{
motion_detected = true;//motion was detected on red pixel
5 } //end if (resultred_pixel > noise_threshold)
result_green_pixel      =      frame_delayed_green_pixel      -
live_green_pixel;//subtract green pixels
if (result_green_pixel > noise_threshold) // test if
green_result is greater than noise threshold
10 {
motion_detected = true;//motion was detected on green pixel
} //end if (result_green_pixel > noise_threshold)
result_blue_pixel      =      frame_delayed_green_pixel      -
live_green_pixel;//subtract green pixels
15 if (result_blue_pixel > noise threshold) //test if blue
result is greater than noise threshold
{
motion_detected = true;
} //end if (result_blue_pixel > noise_threshold)
20 return motion_detected;//motion was detected on blue pixel
} //end of motion detection algorithm

//Procedure for one frame of video, (Executed for each frame
of video)

do//process all pixels for each frame (hd = 777,600 pixels
25 per frame)

//(31 instructions executed for each pixel)

```

```

{
    if (pixel_clock = true) ;//pixel_clock is from the live video
    stream, hd = 46.656 MHz,
    {
//extract red, green, and blue pixels from live video and
5  delayed video pixels
    frame_delayed_pixel = delay_buffer_output_pixel;//get a
    frame delayed video pixel
    live_video_pixel = live_video_input_pixel; // get a live
    video pixel
10  extract_live_red_green_blue (int live_video_pixel) ;//
    extract live color pixels, (9 instructions/pixel)
    extract delayed_red_green_blue (int
    frame_delayed_pixel);//extract delayed color pixels, (9
    instructions/pixel)
15  }//end if (pixel clock = true)
    motion_detect = detect_motion();//call function to detect
    motion (11 instructions/pixel)
    if (motion_detect = true);//motion_detect swaps the
    processing of video outputs 1 and 2
20  {
    output1;//Only moving video is displayed. Non-moving video
    is black.
    output2;//Only non-moving video is displayed. Moving video
    is black.
25  }//end if (motion detect = true)

```

```

if (motion detect = false); //motion_detect swaps the
processing of video outputs 1 and 2
{
output1; //Only non-moving video is displayed. Moving video
5 is black.
output2; //Only moving video is displayed. Non-moving video
is black.
} //end if motion_detect = false)
until (end of frame = true) ; // end of frame is a signal in
10 the video stream

```

[0087] La Figura 7 muestra una ilustración esquemática de un diagrama de una configuración de hardware resultante de una compilación del Ejemplo de Código 1 con un compilador AMPC. Un dispositivo de video 111 tiene dos salidas: una

15 secuencia de píxeles de video vivo 113 y una secuencia de buffer de retardo de cuadro 112. Para salida RGB, cada píxel comprende rojo, verde y azul. El DSP 115 realiza una comparación de la alimentación en vivo y la alimentación diferida, y el resultado se canaliza 117 al LDP 116, que

20 determina si se detecta movimiento. El resultado es emitido por el MBS del RAPC 114. Un único RPAC está configurado para implementar los 3 bloques de procesamiento que se ejecutan en paralelo en cada ciclo de reloj. En comparación, un sistema de procesamiento convencional requiere la ejecución

25 de 37 instrucciones para procesar cada píxel de video para detectar movimiento. La mayoría de estas instrucciones toman

más de 3 ciclos de reloj cuando se ejecutan en un procesador convencional, no reconfigurable y no procesado. Incluso si una instrucción promedio se ejecuta en 3 ciclos de reloj, que es generoso para los no optimizados, a los procesadores de propósito general, sin embargo, les tomaría 111 ciclos de reloj de CPU para procesar cada píxel. A medida que el número de píxeles aumenta en las cámaras modernas, es evidente que los tiempos de ciclo disponibles en los procesadores modernos de núcleo único y múltiple son inadecuados para el trabajo.

10 **[0088]** Por el contrario, la configuración de ejemplo del único procesador RAPC configurado por un compilador AMPC a partir del Ejemplo de código 1 procesa una secuencia continua de píxeles utilizando el reloj de píxeles del video. Tres bloques de procesamiento (DSP, LDP y MBS) se implementan en una configuración de flujo continuo de FPGA con tres ciclos de reloj de latencia, pero cada ciclo de reloj después de llenar el canal (después de los primeros tres ciclos de reloj del reloj de píxel del video) procesa la salida de un píxel, que es un píxel por ciclo de reloj en comparación con un

20 píxel por cada 111 ciclos de reloj. Como resultado, un solo RAPC se desempeña al menos 111 veces más rápido que un solo núcleo de un sistema de procesamiento convencional, es decir, un píxel procesa cada ciclo de reloj en el ICAT en comparación con 37 instrucciones x 3 ciclos de reloj por

25 instrucción o 111 ciclos de reloj por píxel para el procesador convencional. Dado que dos mil (o más)

procesadores RAPC pueden implementarse en un solo chip ICAT, la potencia de procesamiento combinada podría ser al menos 222,000 más rápida que un procesador convencional de un solo núcleo. Los procesadores convencionales actuales están
5 limitados a cuatro núcleos o similares, pero la adición de núcleos a un procesador convencional no es sin sobrecarga adicional. Se pueden agregar muchos más RAPCs que los núcleos de procesamiento convencionales, y cada uno se puede reconfigurar como una canalización sola o junto con otros
10 RAPCs.

[0089] El punto del ejemplo de código 1 y la figura 7 es que la adición de RAPCs es simplemente una cuestión de densidad y tamaño de un chip y se pueden agregar miles a un ASIC sin agregar sobrecarga. Cada RAPC es un procesador
15 paralelo y flexible. Por lo tanto, agregar núcleos, agregar memorias caché y sobreaceleración de procesadores convencionales nunca podría hacer que un procesador convencional se acercara al rendimiento de un solo chip ICAT con un par de docenas de RAPCs. Además, todo el esfuerzo
20 para impulsar los procesadores convencionales da como resultado un calor, costo y tamaño excesivos para un procesador convencional, no reconfigurable y no procesado. Sin mencionar también que estos mismos métodos podrían usarse para aumentar el rendimiento de los RAPC de la arquitectura
25 ICAT. En cualquier caso, agregar RAPC a la arquitectura ICAT siempre mejorará el rendimiento de forma espectacular en

comparación con una arquitectura de procesador convencional comparable, sin la necesidad de que los programadores programen específicamente para la arquitectura ICAT. Este es un resultado sorprendente e inesperado. Toda la atención se

5 centra en obtener más de los procesadores convencionales, mientras que se ha prestado poca atención al agregado de arquitecturas reconfigurables y programables a los procesadores convencionales para mejorar el rendimiento de los procesadores de propósito general.

10 **[0090]** Además, la implementación de la misma solución para el Ejemplo de código 1 en FPGA estándar requeriría algo más que simplemente recompilar un lenguaje de programación de alto nivel estándar, como se proporciona en este ejemplo. Para desarrollar con éxito un multiplicador matricial, PID

15 o cualquier algoritmo complejo, en un FPGA Xilinx, por ejemplo, se requieren las siguientes habilidades: un conocimiento práctico en el diseño de circuitos con lenguajes RTL y Verilog; habilidades avanzadas de arquitectura (procesamiento paralelo, canalización, transmisión de datos,

20 compensaciones de recursos/rendimiento, etc.); experiencia de diseño con una amplia variedad de bloques de construcción de hardware, tales como aritmética, toma de decisiones lógicas, dispositivos de memoria, dispositivos de control, interfaces periféricas, etc.; diseño de software; un

25 conocimiento práctico con varias versiones de lenguajes de programación de nivel superior; un conocimiento práctico con

algoritmos matemáticos utilizados en aplicaciones de monitoreo y control; y un conocimiento de cómo usar las herramientas de software de Xilinx, como compilar el código "C" para el hardware Xilinx; verificar el diseño de hardware y hacer modificaciones de arquitectura si es necesario para cumplir los objetivos de rendimiento; construir el banco de pruebas del código "C"; verificar los resultados de la simulación de hardware contra los resultados del banco de pruebas; e implementar el diseño en hardware y probarlo.

Todo esto hace que un proyecto típico de FPGA sea oportuno y costoso, mucho más allá de la capacidad de una persona que tiene habilidades ordinarias de programación de alto nivel. El estado actual de la tecnología reserva los FPGAs para el procesamiento de nicho donde el rendimiento es primordial y las demoras y los costos de diseño personalizado y la programación son aceptables.

[0091] Por el contrario, cualquier buen programador de lenguaje de alto nivel puede programar la tecnología ICAT, ya que la arquitectura del microprocesador es una arquitectura familiar de uso general. Los RAPCs están configurados por el procesador de propósito general y el AMPC, que usa la estructura estándar de cada RAPC para reconfigurar uno o más RAPCs, en base al código estándar para el procesador frontal, como se ilustra en el diagrama de la Figura 7, por ejemplo. Por lo tanto, la tecnología ICAT, que incluye una pluralidad de RAPCs y un AMPC para

configurar y reconfigurar los RAPCs utilizando una arquitectura de procesador estándar que enfrenta el mundo, es un avance sorprendente e inesperado sobre los procesadores convencionales y cualquier procesador FPGA conocido.

5 **[0092]** La Figura 8 ilustra una aplicación de un microprocesador que combina un núcleo canalizado algorítmico reutilizable con un compilador canalizado de correlación algorítmica. La falta de velocidad de procesamiento de video de los microprocesadores convencionales requiere conjuntos
10 de chips especializados y costosos o procesamiento posterior. Como se muestra en el dibujo, un procesador de propósito general con RAPC y AMPC brinda una solución que procesa millones de píxeles en tiempo real, proporcionando detección de movimiento, compresión de video y velocidades
15 de carga y descarga más rápidas para video desde un chip ICAT de uso general a bordo de un dispositivo electrónico de consumo, por ejemplo.

[0093] Cada RAPC puede comprender un DSP, un LDP y un MBS. Un DSP puede tener una interfaz de configuración para
20 programar los tipos de operaciones requeridas (por ejemplo, enteros y de coma flotante, multiplicar, dividir, sumar, restar, etc.). Un DSP puede tener cuatro entradas para datos de operandos que pueden concatenarse u operarse con varias combinaciones de funciones matemáticas determinadas por los
25 datos de configuración, como se ilustra en la Figura 8. El DSP puede tener un acumulador de 48 bits que se emite como

resultado datos junto con los datos de estado. Los datos de estado incluyen, ejecutar, igual, mayor que y menor que, por ejemplo.

[0094] Un LDP puede tener una interfaz de configuración para programar la tabla de búsqueda, el contador de bucles y el registro constante, por ejemplo. El LDP puede tener un contador de bucle para detectar cuándo se completan los algoritmos iterativos. El LDP puede tener un registro que puede contener datos constantes para la entrada a la tabla de búsqueda. El LDP puede tener un bloque de memoria que se puede usar para realizar funciones. Las funciones LUT pueden incluir una tabla de búsqueda que se puede implementar y acceder secuencialmente utilizando el contador de bucles; una tabla de búsqueda que puede ser implementada y accedida por el estado del DSP, el registro constante o los datos del resultado del DSP para fines de control; y una tabla de búsqueda lógica que puede implementarse y emitir señales lógicas misceláneas para fines de control. El LDP puede pasar datos de resultados desde su entrada a su salida. El LDP puede tener un registro de canalización para los datos de resultado en su salida, por ejemplo. Alternativamente, el LDP puede tener dos registros de canalización con limpieza sincrónica que permite obtener los resultados en su salida.

[0095] Esta descripción detallada proporciona ejemplos que incluyen características y elementos de las reivindicaciones con el fin de permitir que una persona que

posea una experiencia normal en la técnica pueda realizar y usar las invenciones enumeradas en las reivindicaciones. Sin embargo, estos ejemplos no están destinados directamente a limitar el alcance de las reivindicaciones. En cambio, los

5 ejemplos proporcionan características y elementos de las reivindicaciones que, habiendo sido revelados en estas descripciones, reivindicaciones y dibujos, pueden ser alterados y combinados de maneras que son conocidas en la técnica.

10 **[0096]** Por ejemplo, sin limitarse de ningún modo, 3325 RAPCs pueden configurarse en un solo chip Xilinx® Zynq® FPGA, donde Xilinx® y Zync® son marcas comerciales de Xilinx, Inc., funcionando a una tasa de reloj modesta de 100 MHz. En cada reloj, cada uno de los RAPCs puede procesar 1 o 2 operaciones

15 lógicas y una operación matemática. Por lo tanto, esta configuración produce 332 GigaFLOPS. Por ejemplo, esta configuración usa tablas de búsqueda (LUT) para cada una de las cuatro operaciones matemáticas (por ejemplo, sumar, restar, multiplicar, dividir) y cuatro operaciones lógicas

20 (por ejemplo, mayor que, menor que, igual, no igual). El tamaño de memoria LUT estándar es de 512 bytes. Además, se puede proporcionar un LUT "mayor que un valor constante configurable", además de las otras LUT de operación lógica. En un ejemplo, las señales de salida de los LUT se usan para

25 controlar los conmutadores multiplexores de bus para obtener resultados de dirección entre los RAPCs. El compilador AMPC

precompila código fuente de un lenguaje de programa de nivel superior escrito para una arquitectura von Neuman, y el compilador AMPC selecciona LUT para cada operación realizada por un RAPC, generando un procesador que no es von-Neumann
5 a partir del código fuente escrito para la arquitectura von Neumann.

[0097] En comparación con cualquier procesador convencional von Neumann, los 332 GigaFLOPS, donde un GigaFLOPS se define como mil millones de operaciones de punto
10 flotante por segundo, es respetable, especialmente cuando se entiende que esto se obtiene sin ningún requisito especial de enfriamiento para el chip. En comparación, un sistema de procesamiento von Neumann convencional requiere ciclos de búsqueda y ejecución separados para cada operación de
15 matemática, lógica y de ramificación, mientras que los RAPCs no requieren ciclos de recuperación y ejecución separados para cada operación matemática, lógica y de ramificación.

[0098] En un ejemplo, los cálculos muestran que un chip Xilinx® Virtex ZU13, donde Xilinx y Virtex son marcas
20 comerciales de Xilinx, Inc., con una velocidad de reloj de 741 MHz, se puede configurar con 236,250 RAPCs, dando a este chip la capacidad de funcionar a más de 175,000 gigaFLOPS, que es un resultado extraordinario, tanto inesperado como sorprendente para los expertos en la materia. Este resultado
25 es posible porque RAPC no requiere un ciclo de recuperación y ejecución por separado para cada operación de matemática,

lógica y ramificación realizada. Este y otros problemas planteados por la arquitectura von Neumann de procesadores de computadoras de propósito general se resuelven usando RAPCs y las arquitecturas aquí descritas. Los programas
5 escritos para procesadores con una arquitectura von Neumann (es decir, todos los procesadores de propósito general modernos y conocidos) no necesitan ser reescritos para ejecutarse en la arquitectura descrita, un resultado muy sorprendente e inesperado para los expertos en la materia e
10 incluso expertos en esta técnica.

FIG. 1A (TECNICA ANTERIOR)

Cuadro de flujo del compilador convencional

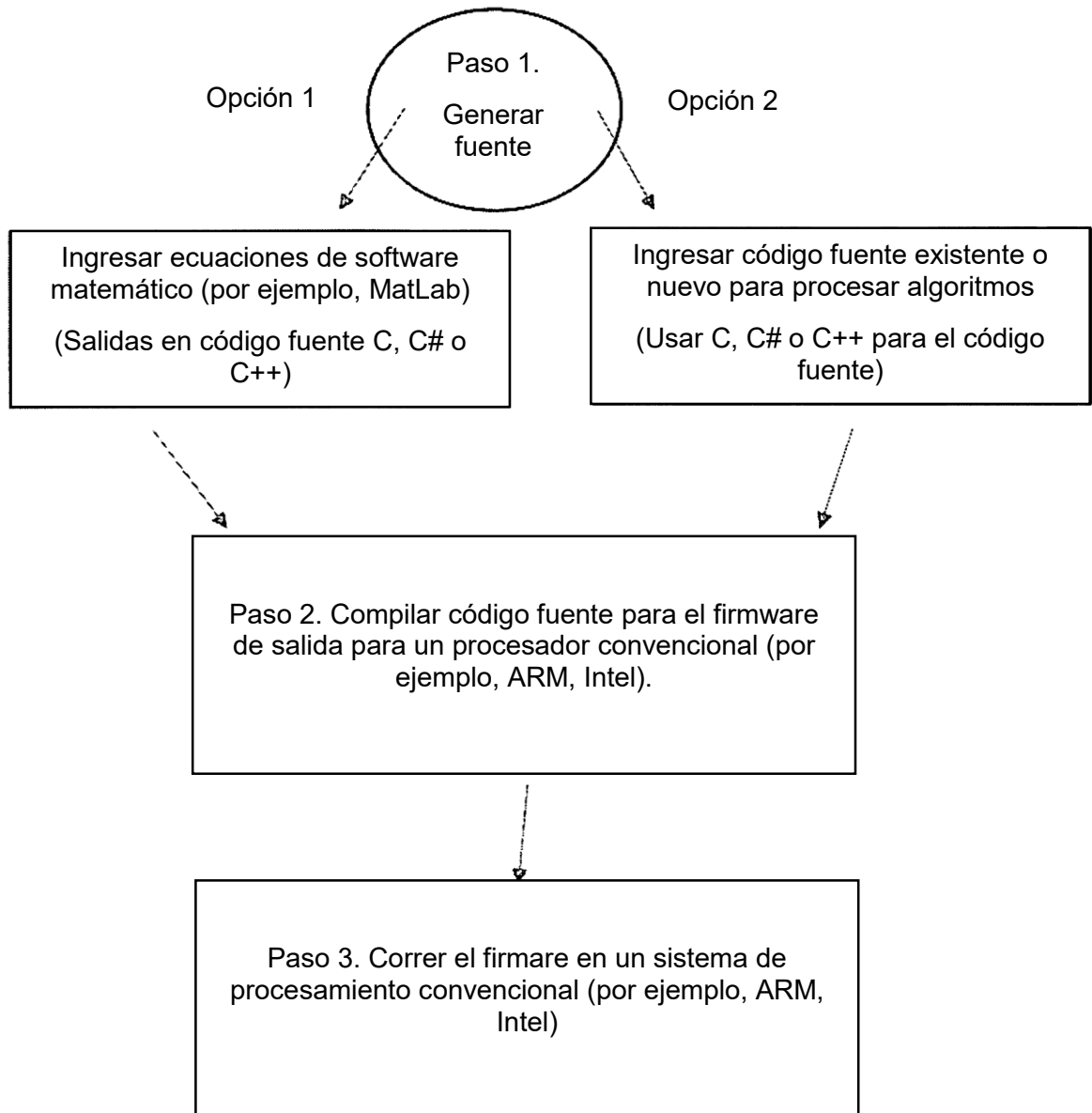
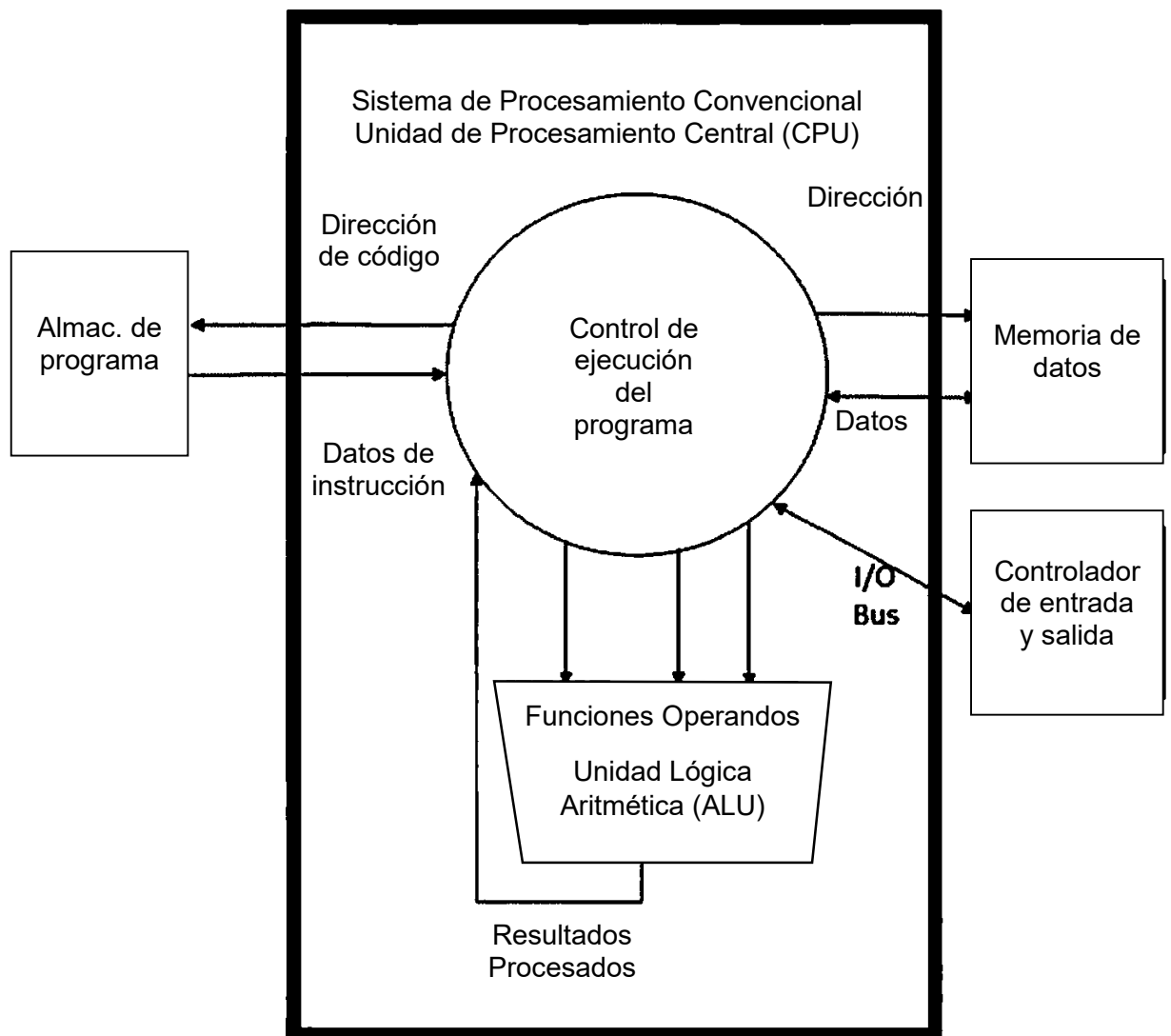


FIG. 1B (TECNICA ANTERIOR)



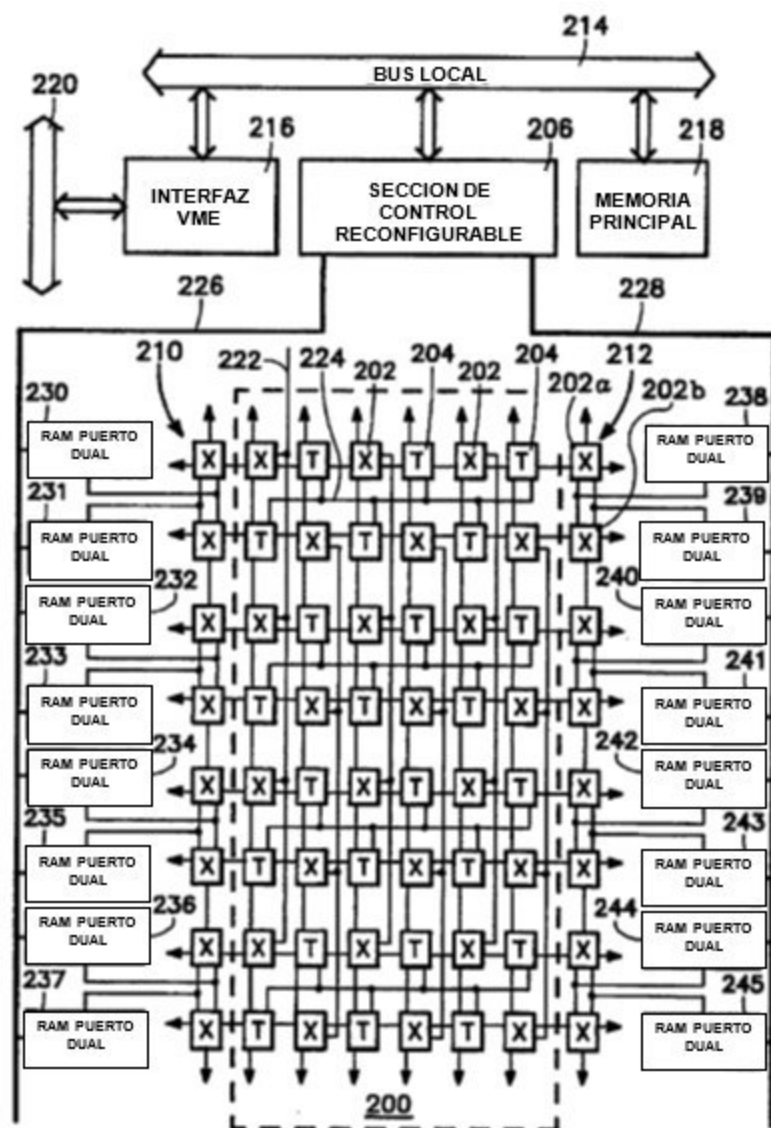
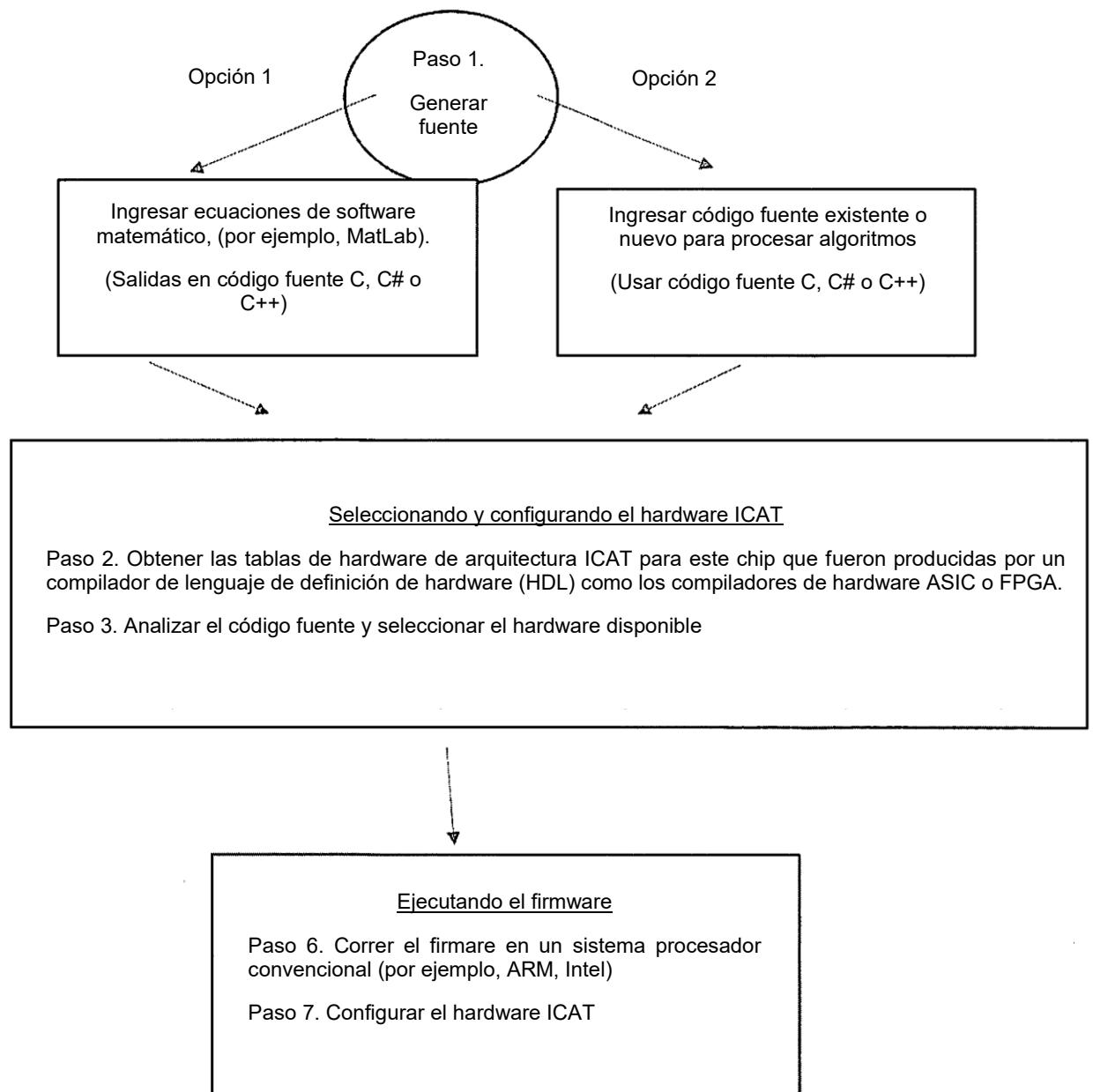


Figura 2
(TECNICA ANTERIOR)

FIG. 3

Cuadro de flujo del compilador AMPC



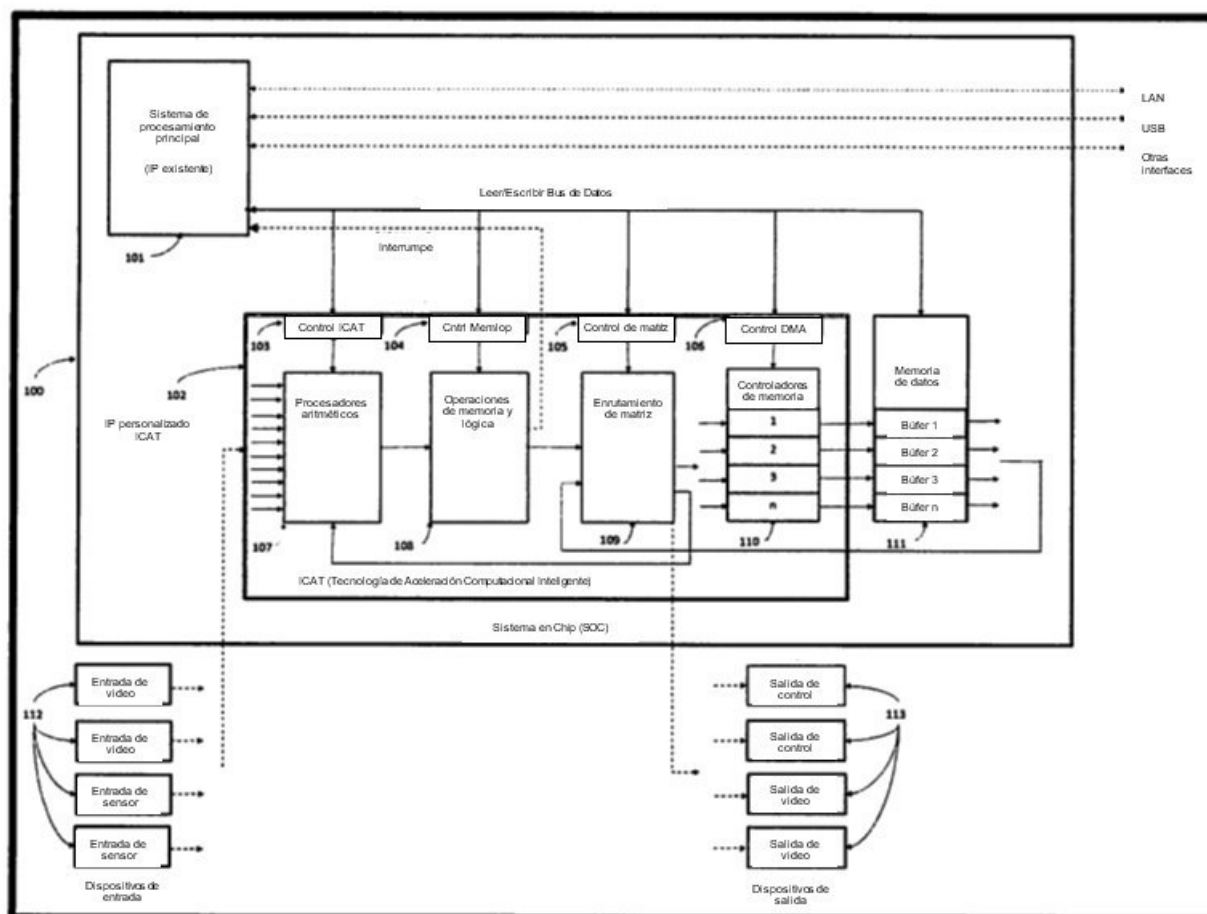


Figura 4

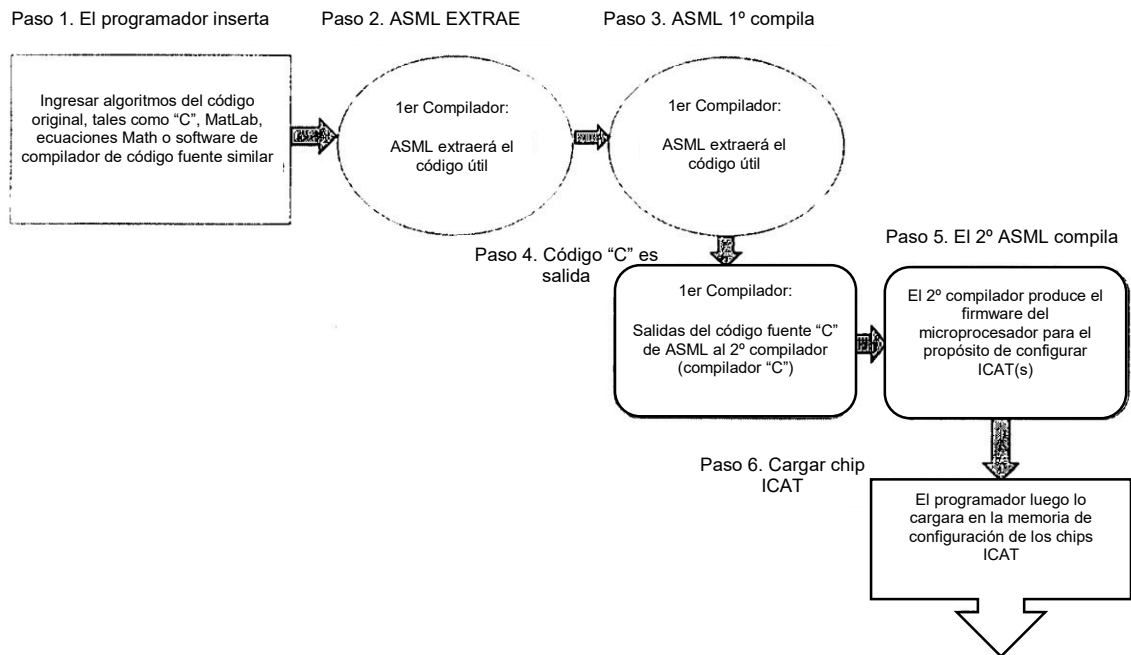


FIG. 5

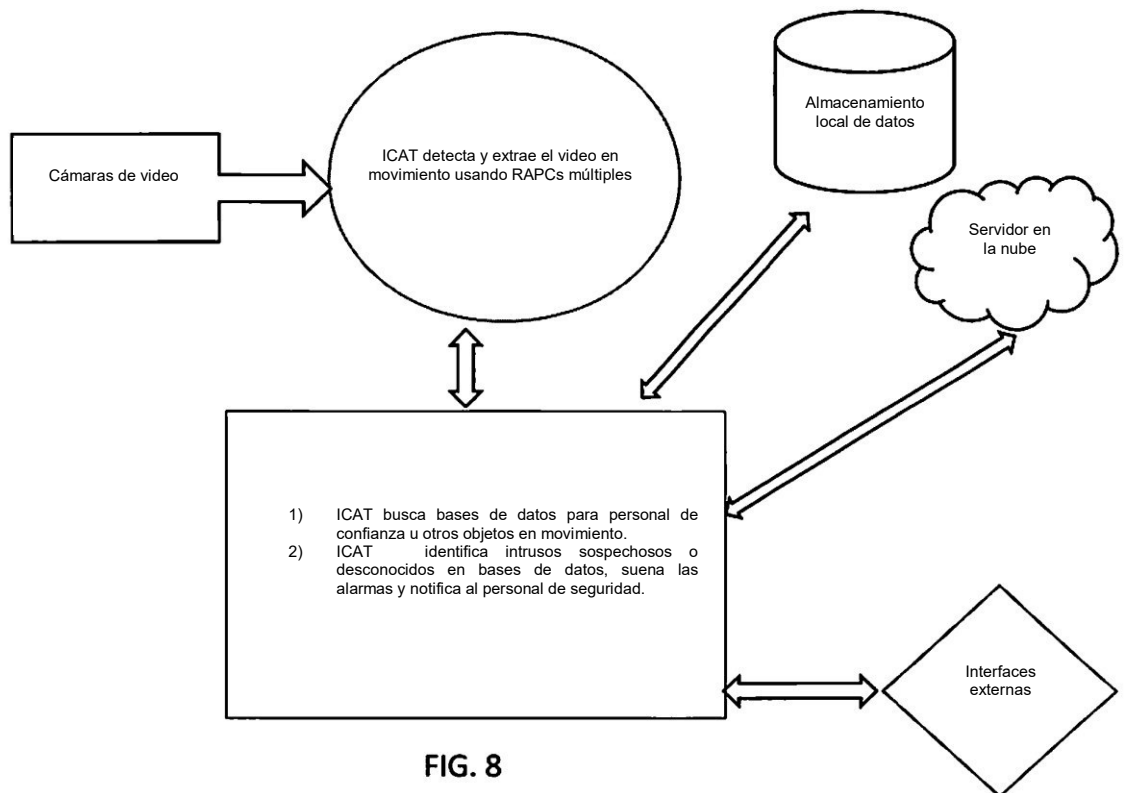


FIG. 8

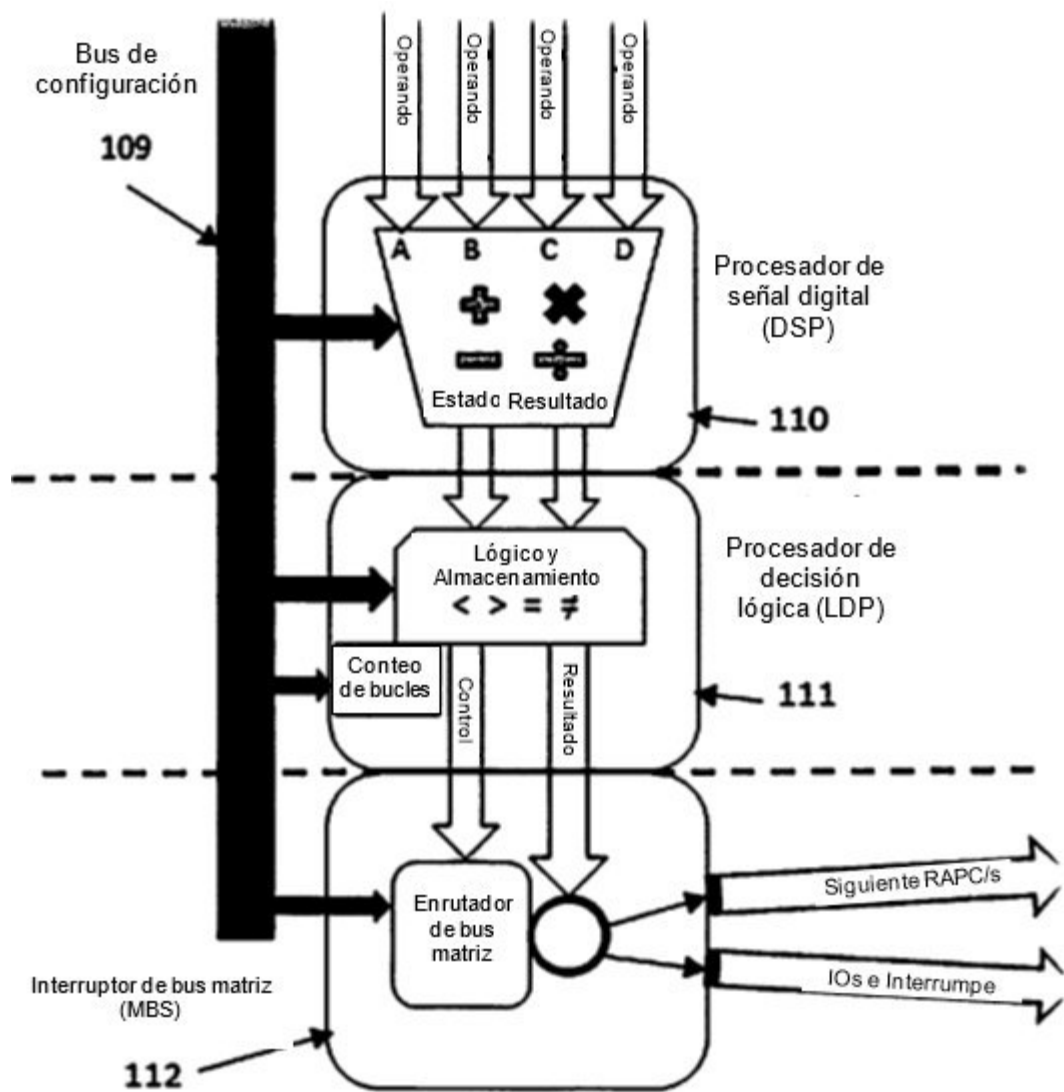
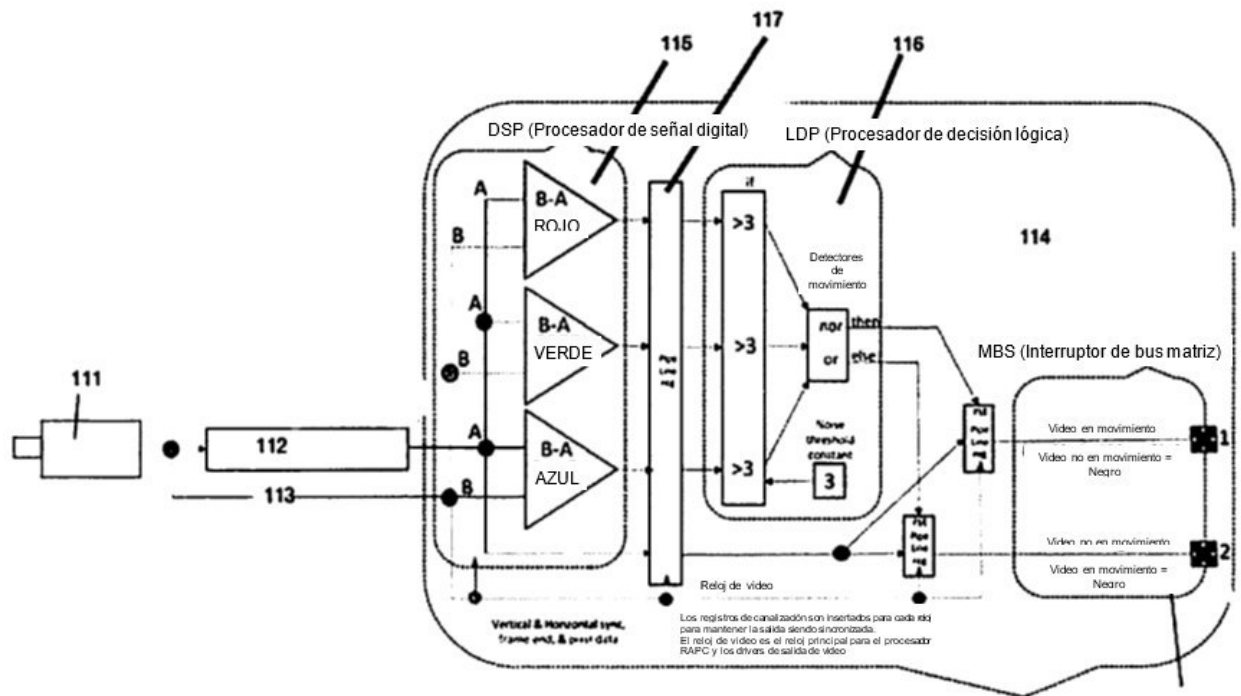


Figura 6



5

Figura 7

RESUMEN

Un compilador canalizado de correlación algorítmica y un núcleo canalizado algorítmico reutilizable comprenden un sistema. El núcleo canalizado algorítmico reutilizable es un

5 núcleo de procesamiento reconfigurable con una estructura canalizada que comprende un procesador con una interfaz de configuración para programar cualquiera de una pluralidad de operaciones determinadas por datos de configuración, un procesador de decisión lógica para programar una tabla de

10 consulta, un contador de bucle y un registro constante y un bloque de memoria. Esto se puede usar para realizar funciones. Un circuito programable y reconfigurable enruta los datos y los resultados de un núcleo a otro núcleo y/o 10 controladores y/o generador de interrupción, según se

15 requiera para completar un algoritmo sin intervención adicional de un procesador central o periférico durante el procesamiento de un algoritmo.



SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO

NIT: 800.176.089-2

COMPROBANTE DE RECAUDO SISTEMA SIPI

Radicado: NC2018/0008835
Recibo: 20-32002
Fecha: 13/05/2020

CAROLINA MERCEDES DAZA MONTALVO
CC: 66.783.790
Pago PSE No: 398704
CUS: 624751770

Patente PCT
PROCESADOR CON NÚCLEO CANALIZADO ALGORITMICO RECONFIGURABLE Y COMPILADOR
CANALIZADO DE CORRELACION ALGORITMICA

Detalle del pago

Id.	Rentístico	Concepto y Servicio	Cant.	Val. Unit.	Subtotal
1	50005-01-01	Solicitud examen de patentabilidad Patente PCT (DESCUENTO INTERNET)	1	\$ 1,133,000	\$ 1,133,000
TOTAL					\$ 1,133,000